МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних систем та мереж

"ДОПУСТИТИ ДО ЗАХИСТУ"
Завідувач кафедри
__________________ Жуков І.А.
"_____" _____________ 2020 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

випускника освітнього ступеня "МАГІСТР"

спеціальності 123 «Комп'ютерна інженерія»

освітньо-професійної програми «Комп'ютерні системи та мережі»

на тему: **«Система безперервної програмної обробки з використанням хмарних технологій»**

Виконав: __________________________ Бондаренко К.І.

Керівник: __________________________ Тележко І.В.

Нормоконтролер: __________________ Надточій В.І.

Засвідчую, що у дипломній роботі немає запозичень з праць інших авторів без відповідних посилань

Київ 2020

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL AVIATION UNIVERSITY

Faculty of Cybersecurity, Computer and Software Engineering

Computer Systems and Networks Department

PERMISSION TO DEFEND GRANTED

The Head of the Department

_____________________ Zhukov I.A.

"_____"____________________2020

MASTER'S DEGREE THESIS

(EXPLANATORY NOTE)

Specialty: 123 Computer Engineering

Educational-Professional Program: Computer Systems and Networks

Topic: System of continuous software development using cloud technologies

Completed by: ____ Bondarenko K.I.

Supervisor: ____ Teleshko I.V.

Standards Inspector_____________________________________ Nadtochiy V.I.

Kyiv 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп’ютерної та програмної інженерії

Кафедра Комп’ютерних систем та мереж

Освітньо-кваліфікаційний рівень “Магістр”

Спеціальність 123 “Комп’ютерна інженерія”

Спеціалізація 123.01 “Комп’ютерні системі та мережі”

ЗАТВЕРДЖУЮ
Завідувач кафедри
________________Жуков І.А.

« ____ » ____________ 2020 р.

З А В Д А Н Н Я
на виконання дипломної роботи

Бондаренка Костянтина Ігоровича
(прізвище, ім'я, по батькові)

1.Тема дипломної роботи “ Система безперервної програмної обробки з використанням хмарних технологій”
затверджена наказом ректора від "25" вересня 2020 року № 1793/ст.
2.Термін виконання проекту (роботи): з 01.10.2020 до 12.12.2020
3. Вхідні дані до роботи (проекту): принципи формування хмарних середовищ, розмежування платформ розробки, централізоване зберігання корпоративних даних
4. Зміст пояснювальної записки:
Вступ, огляд теми, огляд існуючих платформ розробки програмного забезпечення з використанням хмарних технологій, розробка хмарної комп’ютерної мережі, висновки по роботі
5. Перелік обов'язкового графічного (ілюстративного) матеріалу:
Матеріали представленні у вигляді презентації в *Power Point.*

6. Календарний план-графік

№ п/п	Етапи виконання дипломного проекту	Термін виконання етапів	Примітка
1	Узгодження технічного завдання з керівником проекту	01.10.20	
2	Підбір та вивчення науково-технічної літератури за темою дипломної роботи	06.10.20 – 07.10.20	
3	Опрацювання теоретичного матеріалу	08.10.20 – 15.10.20	
4	Аналіз стрімко-розвивающихся хмарних технологій для забезпечення системи безбеперервної обробки	16.10.20 – 25.10.20	
5	Проектування системи безбеперервної обробки та тестування хмарних рішень	26.10.20 – 25.11.20	
6	Оформлення пояснювальної записки	26.11.20 – 05.12.20	
7	Оформлення графічних матеріалів проекту та представлення роботи на кафедру	06.12.20 – 12.12.20	

7. Дата видачі завдання «01» жовтня 2020 р.

Керівник дипломної роботи________________________ Тележко І.В.
(підпис)

Завдання прийняв до виконання______________________ Бондаренко К.І.
(підпис студента)

NATIONAL AVIATION UNIVERSITY

Faculty of Cybersecurity, Computer and Software engineering

Department: Computer Systems and Networks

Educational and Qualifications level: Master Degree

The Specialty 123 “Computer Engineering”

The Specialization 123.01 “Computer Systems and Networks”

APPROVED BY

The Head of the Department

____________________ Zhukov I.A.

“_____”____________________2020

Graduate Student’s Degree Project Assignment

Bondarenko Kostiantyn Ihorovych

1. The Project topic: “System of continuous software development using cloud technologies”

Approved by the Rector’s order of 25.09. 2020 № 1793/st.

2. The Thesis to be completed between 01.10.2020 and 12.12.2020

3. Initial data for the project (thesis): principles of formation of cloud environments, delimitation of development platforms, centralized storage of corporate data

4. The content of the explanatory note (the list of problems to be considered):

Introduction, review of the topic, review of existing software development platforms using cloud technologies, development of a cloud computer network, conclusions on the work

5. The list of mandatory graphic materials:

Materials representation in the form of Power Point Presentation

6. Timetable

#	**Completion Stages of Degree Project**	**Stage Completion Dates**	**Remarks**
1	Technical task coordination with the supervisor	01.10.20	
2	Selection and study of scientific and technical literature on the topic of the thesis	06.10.20 – 07.10.20	
3	Elaboration of theoretical material	08.10.20 – 15.10.20	
4	Analysis of rapidly evolving cloud technologies to ensure a system of continuous processing	16.10.20 – 25.10.20	
5	Design of a system for continuous processing and testing of cloud solutions	26.10.20 – 25.11.20	
6	Making an explanatory note	26.11.20 – 05.12.20	
7	Design of graphic materials of the project and presentation of work to the department	06.12.20 – 12.12.20	

7. Assignment issue date: 01.10.20

Diploma Thesis Supervisor ____________________________ Teleshko I.V.
(Signature)

Assignment accepted for completion ______________________ Bondarenko K.I.
(Signature)

ABSTRACT

The Explanatory Note to the Master's Degree Thesis "System of continuous software development using cloud technologies": 80 pages, 30 figures, 24 references.

CLOUD TECHNOLOGY, CONTINIOUS INTEGRATION AND DELIVERY, CLOUD NETWORK, SOFTWARE.

The Goal of the Master's Degree Thesis – create the continuous integration project in cloud platform for group of developers.

Main Tasks – to analyze the process of speeding up assembly and improving the quality of the final product of development.

The Designing Object of Project – automatization of processes of the identify potential defects and solve integration problems.

The Subject of Project – continiious integration process using cloud computers. Set up automatic assembly of applications for testing in QA and demonstration to the customer.

Practical Usage – automation of integration and delivery helped to significantly optimize all routine assembly processes using cloud technologies.

CONTENT

LIST OF SYMBOLS, ABBREVEATIONS, TERMS

ASP	–	Active Server Pages
IaaS	–	Infrastructure as a service
SaaS	–	Software as a service
PaaS	–	Platform as a service
WaaS	–	Platform as a service
SETI	–	Search for extraterrestrial intelligence
EC2	–	Elastic computer cloud
AWS	–	Amazon Web Services
API	–	Application programming interface
KVM	–	Kernel Virtual Machine
RDS	–	Relational Database Service
S3	–	Simple Storage Service
GSP	–	Google Cloud Platform
CEP	–	Complex Event Processor
QA	–	Quality assurance
xCAT	–	Cloud Administration Toolkit
VPC	–	Virtual Private Cloud
SSH	–	Secure Shell
NIST	–	The US National Institute of Standards and Technology
VLAN	–	Virtual Local Area Network
DNS	–	The Domain Name System
HTTP	–	The Hypertext Transfer Protocol
DHCP	–	The Dynamic Host Configuration Protocol
TFTP	–	The Trivial File Transfer Protocol
NFS	–	Network File System
SQL	–	Structured Query Language
NC	–	Node Controller

CC	–	The Cluster Controller
CLC	–	Cloud Controller
SC	–	Storage Controller
CI/CD	–	Continious Integration and Continious Delivery
AMQP	–	The Advanced Message Queuing Protocol
RDS	–	Relational Database Service
SSD	–	A solid-state drive
OCCI	–	Open Cloud Computing Interface
DBMS	–	A database management system
ELB	–	Elastic Load Balancing
HVM	–	Hardware Virtual Machine
AZ	–	Availability zone
AMI	–	Amazon Machine Image
DEVOPS	–	Development and Operations
RPM	–	Red-Hat Package Manager
JSON	–	JavaScript Object Notation

INTRODUCTION

Actuality of theme. In the modern economy, the use of digital tools in business decisions plays a defining role. With the increasing complexity of high-tech platforms, the continuity of critical IT systems is becoming an important factor. This trend also affected software development. Nowadays, a very high interest in the tasks of optimization, saving time and money for large and small businesses is determined by the need to automate both software processing and continuous integration in real time in cloud systems. Therefore, now the search and implementation of effective methods and principles of continuous software processing using cloud computing systems.

I believe that more promising directions for solving this problem is based on the use of cloud platforms and services, as the most advanced solution to the problems of ensuring uninterrupted integration and delivery to ensure processing.

Currently, a huge number of possible uninterrupted processes are proposed for solving software processing problems. Significant difficulties arise when migrating from one platform to another, using conflicting services and inaccessible storage.

This problem is solved by choosing appropriate software with selected services and testing method. Analysis shows that there is still no model that could be reliable and work. The problem is relatively resolved when using the services of some platforms and delivery using load balancers.

However, as before, there are still some difficulties with secure access to instances for testing and development long processing times. A promising way to overcome these problems is to configure individual local area networks to block access only to certain ports through different types of connections.

The purpose of the thesis is projectionng, testing and executing pipelines for continuous processing.

Research methods. The thesis covers diverse approaches to solve continuous delivery tasks. The main attention is paid to the automatization of the processes of building applications and their non-stop delivery to the necessary repositories for further processing.

Scientific novelty of the obtained results. Organization of interaction between project teams excluding the possibility of human error. Engineers,analysts, development teams, and other functional divisions began to work in a single ecosystem.

The practical significance of the thesis results lies in the possibility of implementing the results obtained for the effective use of the resources that are available, as well as accelerating the processing time and data availability for analysis and further development. In addition, it should be noted that the proposed architecture of the model can be improved to obtain better results in terms of the time used for each stage.

PART 1
DIFFERENT CLOUD OPTIONS AS AN OPPORTUNITY TO IMPROVE EFFICIENCY AND SPEED UP THE DEPLOYMENT IN OUR COMPANY

1.1. Analysis of cloud computing architecture, types of services, main components and usage models for optimised our infrastructure

The most popular topic in the field of information technology is "cloud services". Not to describe with the pen everything that is said about the cloud industry, even more talk about it at various scientific conferences. And things are still there, a large audience of people, still unaware of the "know-how" of the "cloud". Let's pump a little theory into the knowledge barrel.

Cloud technology is a remote service on the Internet where a variety of hardware services are provided to registered users of the service.

Most computer users have email. Let's say you signed up for the gmail mail service. It allows you to get an e-mail address and correspond for free. These are the cloud services of the Gmail service [1].

Processing the image in Photoshop, you are not even close to the cloud industry, all the work takes place on your laptop. You are tired of fiddling with Photoshop and you upload a picture to the Picasa resource, it turns out that you are already working not for yourself. This will already mean that you are working in the "cloud", which is controlled from a laptop. A little oversimplified, but understandable.

All work that happens directly on your laptop is not considered work in the cloud, but if the user's actions take place on a remote service, then it will be considered that the user has used cloud technology services. The cloud industry can be thought of as a software help to the user for solving their problems. The implementation of layers of cloud computing presented in Fig 1.1.

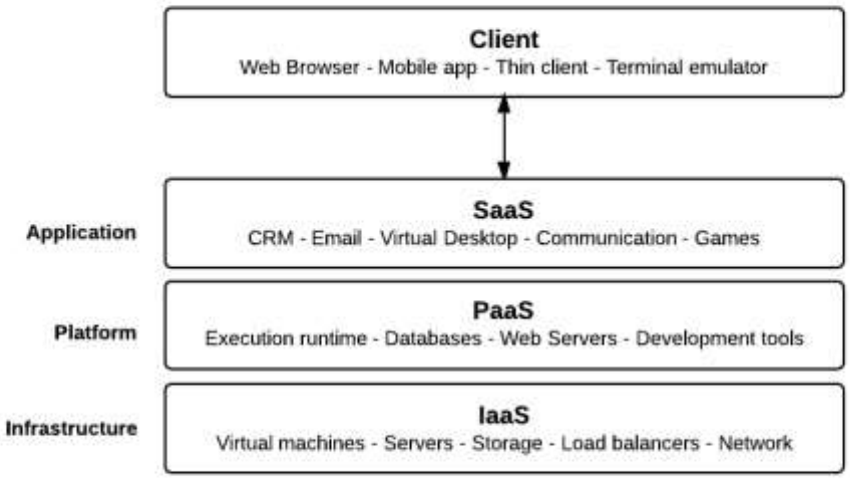

Client
Web Browser - Mobile app - Thin client - Terminal emulator
Application
SaaS
CRM - Email - Virtual Desktop - Communication - Games
Platform
PaaS
Execution runtime - Databases - Web Servers - Development tools
Infrastructure
IaaS
Virtual machines - Servers - Storage - Load balancers - Network

Fig. 1.1. Layers of cloud computing

All major providers provide service in infrastructure as a service - AWS Lambda, Azure Functions, Google Cloud Functions, and IBM OpenWhisk. From the user's point of view, cloud computing makes it possible to receive computing resources over the network from an external provider in the form of a service, payment for which is made depending on the amount of consumed resources, that is, in much the same way as it happens with water or electricity services. At the same time, the amount of computing resources - a virtual computer that the user receives at his disposal - quickly adapts to the current user requests. Convenience of access to the service is provided by the support of a wide range of terminal devices: personal computers, mobile phones, internet tablets [2].

The US National Institute of Standards and Technology (NIST) identifies the following mandatory characteristics of cloud computing:

– self-service on demand - the consumer independently chooses which set of computing resources he will use, and can, if necessary, quickly change this set without consulting the service provider;

– universal access over the network - services are available over the data transmission network regardless of which terminal device the consumer uses;

– resource pooling - the service provider combines the computing resources at its disposal into a single pool (pool) for dynamic redistribution of these resources between consumers; at the same time, consumers control only the basic parameters of the service (for example, data volume, access speed), and the actual distribution of the provided resources is carried out by the supplier;

– instant elasticity (flexibility) - the computing power provided to the user can be promptly increased or decreased automatically, based on the user's needs;

– consumption metering - the service provider automatically calculates the consumed resources (the amount of stored data, the amount of data transferred, the number of users, the number of transactions, etc.), and on this basis estimates the volume of services provided to consumers.

From the point of view of the service provider, the most important part of cloud computing is a data center or data center, which contains:

– information infrastructure for processing and storing information - the main functions of the data center;

– telecommunication infrastructure for data transmission within the data center, as well as between the data center and users;

– engineering infrastructure that ensures normal functioning;

– main systems of the center (air conditioning, uninterruptible power supply, fire and security alarm, etc.).

The key concepts associated with the implementation of "clouds" are scalability and virtualization. Scalability is the ability of a computing system to increase its performance to cope with an increase in workload. Virtualization abstracts cloud computing from the underlying hardware and software infrastructure. Virtualized resources are provided to the user through specific interfaces (APIs or services). This architecture provides the scalability and flexibility of the physical layer of the cloud, isolating changes in the data center from the impact on the end user. A cloud computing service consumer can save on building their own data centers, purchasing appropriate equipment and hiring service personnel. For a data center owner, the benefit of cloud computing is more efficient use of the center's resources. Cloud computing is the result of a long evolution of a range of information technologies. For example, virtualization was first used on IBM mainframes back in the mid-1960s. Computing infrastructure has been provided as a service long before the advent of cloud computing. This approach was called "utility computing" - a term that is now widely used to describe the infrastructure layer of cloud systems. Grid technologies that implement distributed computing on remote computers became famous in the late 1990s thanks to the SETI @ home project for the search for extraterrestrial intelligence.

The beginning of large-scale provision of services for accessing computing resources over the Internet dates back to August 2006, when Amazon.com launched the Elastic Compute Cloud service. The term "cloud" owes its appearance to diagrams depicting the interaction of users with the Internet, in which the latter was represented as a cloud, as a kind of complex infrastructure behind which all the technical details are hidden. The advantage of FaaS applications is that they do not consume resources until the configured code trigger fires. This reduces costs.

IaaS - Infrastructure-as-a-Service provides a virtualized environment based on several servers combined into clusters. In fact, in this option you get a whole infrastructure for use and you can customize it yourself as you want. You can create workplaces, file storages, connect additional servers and develop your own software. If the first two types of services, as a rule, are services for everyone, then IaaS is, to a greater extent, a business solution.

PaaS - Platform as a Service or simplifies application deployment and management while hiding server management, load balancing, DNS, and more. Therefore, there is no need to hire engineers to maintain infrastructure. This allows developers to focus more on development and deployment issues. Companies use software-centric architectures and microservices because they offer the ability to automatically deploy and test code, and scale based on load.

Software as a service (SaaS) is the latest layer of cloud computing that complements PaaS most often, as seen in the diagram at the beginning of this article. It is a fully functional user application that performs specific functions, such as working with images or sound. The most popular form of payment in this segment is subscription.

In the case of SaaS, the responsibility of the cloud provider is transferred to the issues of application configuration, monitoring and backup. Therefore, such a model of work does not require the presence of a technical specialist in the organization's team - everything is done by the provider.

Thus, the higher-level model you plan to use, the less IT competency is required from the team. The converse is also true - the lower your company's IT maturity level, the higher-level model you will need.

SaaS - In the application-as-a-service model, the user only has access to a specific application. The user has no access to either the API or the application code. SaaS applications run on the server of the SaaS provider, and users access them through an Internet browser. The user does not buy a SaaS application, but rents it - he pays a certain amount per month for its use. Thus, an economic effect is achieved, which is considered one of the main advantages of SaaS [3].

SaaS provider takes care of the application's performance, provides technical support to users, and installs updates on its own.

Depending on the audience that has access to the service, the following types of "clouds" can be distinguished:

– private cloud - an infrastructure designed for the use of cloud computing on the scale of one organization;

– community cloud - a type of cloud infrastructure that is designed for the exclusive use of cloud computing resources by a specific community of users solving common problems;

– public or public cloud - an infrastructure designed for the free use of cloud computing by the general public. Such a cloud can be owned, operated and operated by commercial, academic and government organizations and physically exists in the jurisdiction of the owner - the service provider;

– hybrid cloud is a combination of different types of cloud infrastructures. For example, a hybrid cloud can combine;

– private and public clouds, allowing organizations to process sensitive data within their private cloud and transfer other data for processing to the public cloud. The clouds that make up a hybrid cloud remain unique entities linked by standardized or proprietary data and application technologies.

The simplest from a technical point of view is a private cloud. It may not have an API because all of its resources are used within the same organization. The most difficult thing is to implement hybrid clouds, since in this case there is a need to monitor internal and external computing infrastructure, complicate security policies, standardize interaction between heterogeneous cloud infrastructures, etc. As part of the cloud platform, several main components can be distinguished. The core of the platform: the environment and a set of utilities that provide delivery, development and integration of cloud services. The choice of one or another kernel imposes certain restrictions on the methods of developing and delivering an application. For example, you may need to use only the programming languages and development tools supported by this platform. The interface through which the user interacts with the cloud. The most commonly used web interface and various APIs (for example, EC2 or OCCI).

Data store. Modern cloud systems can store huge amounts of user data (we are talking about tens and hundreds of petabytes) and these volumes are constantly growing. In such a situation, classical relational databases no longer give satisfactory results in terms of processing speed. Moreover, cloud platforms often have to deal with related data structures (graphs, trees), which causes additional difficulties when using a relational approach. In this regard, in the last decade, NoSQL DBMSs (columnar DBMSs such as Hadoop; document-oriented DBMSs like CouchDB and MongoDB), as well as alternative approaches to processing extremely large amounts of information, have been actively developing.

User management. Information about the main consumers of cloud resources is used to optimize and fine-tune the cloud for their tasks. First of all, it is necessary to ensure transparent user authorization in all services of the cloud platform. In addition, most applications require the ability to distinguish users from each other in order to provide them with relevant information. At the same time, due to the distributed nature of cloud services, it is necessary to ensure the highest level of security when working with user information.

Monitoring and support of running applications. Administering a running application can be daunting given the large number of individual services that make up a cloud application. In this regard, it is necessary to ensure the integration of administration and service management processes, as well as user tasks in the form of a single "service control center".

1.2. Analysis of free cloud services, hardware, software and organisational features of their implementation

The modern stage of development of cloud technologies began with the launch in by Amazon.com of the cloud computing service Elastic Compute Cloud (EC2) and the online file storage Simple Storage Service (S3). Currently, Amazon Web Services (AWS) integrates more than seventy cloud services, including: data storage, virtual server rental, provision of computing power, etc., and EC2 has become the de facto cloud computing standard. Following Amazon.com, Google and Microsoft offered their cloud platforms. However, starting with proprietary solutions, free and open-source cloud platforms began to appear as competition increased in the market.

The first such platform to achieve commercial success was the Eucalyptus IaaS system. It still remains one of the most popular tools for creating "clouds". However, as technology evolved and user requirements grew, Eucalyptus was criticized for its lack of scalability. Other projects appeared on the market open source, designed to solve this problem: OpenStack, CloudStack, OpenNebula, etc. All of these platforms belong to the IaaS segment. MaaS platforms developed in parallel with them. So, one of the oldest tools of this type - xCAT, having appeared as a tool for managing high-performance computing clusters, later began to support virtualization and cloud management technologies. Open source PaaS systems appeared somewhat later. The first of these systems to become widespread was the Cloud Foundry platform from VMware, one of the leaders in the development of virtualization technologies.

At present, MaaS platforms are understood to mean about two dozen programs with very different functions. Since the term MaaS itself appeared in 2012, in connection with the development of the Juju platform by Canonical, the discussion of the classification of such platforms is still far from complete.

We will understand by MaaS platforms tools that originate from computing cluster management programs, and consider one of the most widespread platforms of this kind - xCAT, the first version of which was released by IBM back in 1999.

xCAT (Extreme Cluster / Cloud Administration Toolkit) is a tool for managing high performance computing clusters, Grid systems, and clouds.

We can highlight Key features of xCAT below. First of all, allows you to run operating systems on physical and virtual machines: RHEL, CentOS, Fedora, SLES, Ubuntu, AIX, Windows, VMware, KVM, PowerVM, PowerKVM, zVM. It provides scripts for installation, stateless, statelite, iSCSI, or for cloning, supports remote control systems: lights-out, console and distributed shells. With a help of it, we have the opportunity to configuration and management of nodes: DNS, HTTP, DHCP, TFTP, NFS. xCat supports installation to hard disk, startup without disk (diskless), startup without disk with the least bit state (statelite), installation on iSCSI (no special hardware required) and support for hypervisors (VMWare, KVM, Xen, PowerVM, ZVM) and technologies Docker [4].

The system is implemented in Perl and distributed under the Eclipse Public License.A significant advantage of xCAT is its good scalability: the developers declare support for clusters consisting of a thousand or more nodes. In addition, since the development of xCAT is coordinated by IBM, this software has advantages when installed on equipment manufactured by that company over third-party software. Note that IBM's share of the mainframe market is about 90%.

Simplifying somewhat, the work of the IaaS platform looks like this. The user makes a request to the master server to allocate a new virtual machine with the specified OS and other parameters. The server processes the request and selects the least loaded machine. There are various algorithms for selecting a node. For example, a request can be sent to the first of the nodes with available free resources, until it is full (Greedy mode), or several nodes in turn (Round-robin mode). After the machine is selected, the server provides it with an image of the specified OS and gives an indication to start the system. The machine starts the image, and the user is returned the IP address of his virtual server. This is roughly how EC2 infrastructure works.

AWS services have grown in popularity, leading to a number of open source service implementations that are EC2 compliant. The most common among such platforms is the Eucalyptus platform [5].

The first version of Eucalyptus (Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems) was released by the company of the same name in 2008. The platform is currently owned by Hewlett-Packard, and its full name is HPE Helion Eucalyptus [5].

Eucalyptus is an infrastructure for implementing a cloud computing model of the IaaS level, the distinctive features of which are:

– EC2 and S3 compliant interface (web services and Query / REST interface);

– support for Xen, KVM and VMware ESX / ESXi hypervisors;

– support for most Linux distributions;

– secure interaction of components using SOAP and WS-security;

– availability of advanced cloud administration tools for system management and user accounting;

– the possibility of combining many clusters, each of which is located in a separate network segment, into a single "cloud".

The Eucalyptus platform has five main components in Fig.1.2:

1. Node Controller (NC). Runs on each node that makes up the "cloud", and is responsible for starting, running and stopping virtual machines;

2. The Cluster Controller (CC) manages the node controllers and decides on which nodes the virtual machines will run;

3. Cloud Controller (CLC). Installed on a machine that has access to an external network, it acts as the head interface for accessing the cloud. Processes user requests to start virtual machines and collects data on node load from cluster controllers;

4. Storage Controller (SC). Interacts with cluster and node controllers and manages the storage of volumes and their snapshots within a specific cluster. If at the same time it is required to write data to a memory area lying outside the cluster, then such data is written to Walrus, which is accessible from any cloud cluster. This controller is analogous to the AWS Elastic Block Store;

5. Walrus is a data warehouse accessible via ReSTful and SOAP API, and is a loose analogue of the S3 data storage service. Walrus allows you to store virtual machine images, volume snapshots, application data, and more [6].

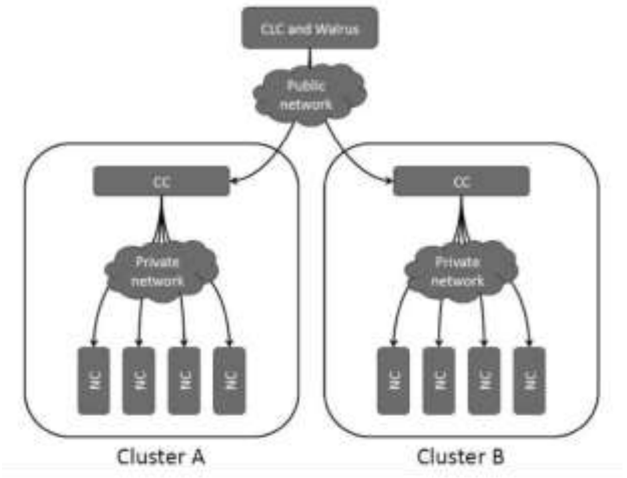


Fig. 1.2. Hierarchical structure of the Eucalyptus platform

Unlike systems such as CloudStack and OpenStack, the Eucalyptus platform is primarily aimed at organizing the operation of private and hybrid clouds. The creation of public clouds in which the access of "outside" users is organized goes beyond the scope of Eucalyptus. The Eucalyptus source code is licensed under the GNU GPL version 3.

Work on the OpenStack project began in the summer of 2010 at the initiative of NASA and hosting provider Rackspace, who made available the source code for their cloud platforms. According to the initiators of the project, OpenStack should eventually become the standard open cloud platform [7].

Currently, OpenStack is a set of 17 free software projects that can be used to create infrastructure cloud services, both private and public. All projects of the complex are distributed under the Apache License. The main component of OpenStack is OpenStack Compute or Nova, the controller that controls the operation of virtual machines. Nova processes requests to create virtual machines, connects them to the outside world, monitors the health and load distribution of physical machines and communication channels, responds to failures, etc. Nova is written in the Python programming language and relies on the AMQP messaging protocol. OpenStack consists of about two dozen separate components, among which the following can be distinguished as the main ones:

1. the cloud controller (Cloud Controller) monitors the state of the system and acts as a connecting link for all other components of the system;

2. the API server implements a web interface that allows you to manage the cloud controller;

3. The Compute Controller is responsible for launching virtual machines and connecting them to the rest of the cloud infrastructure;

4. storage (Object Store) provides a storage service compatible with S3;

5. Auth Manager provides user authentication and authorization;

6. the volume controller (Volume Controller) allows you to connect virtual storage devices to virtual machines;

7. Network Controller creates virtual networks, allowing virtual machines to communicate with each other and the external network;

8. The Scheduler is responsible for choosing the right compute controller to run the new virtual machine.

Processing user requests in OpenStack looks like this: the user sends a request to create a virtual machine to the API server. After the user is authenticated and authorized, the API server parses the request and sends it to the cloud controller, which initiates three new requests: the network controller, the storage, and the scheduler. The network controller allocates an IP address for the new virtual machine (VM) and returns it to the cloud controller. The network storage searches for a suitable hard disk image for the future VM, the address of which is returned to the cloud controller. Now, having everything it needs to create a new VM, the cloud controller sends a request to the scheduler, which chooses the most appropriate compute controller and issues it a request to create a VM. After the new VM is up and running, the cloud controller shuts down and informs the API server of the success of the entire operation [8].

Now the user can connect to the VM, and all the activities to maintain it in working order, allocate disk space, routing network packets, etc. the system takes over.

Although the developers of OpenStack, unlike Eucalyptus, do not set the goal of achieving interface compatibility with other cloud platforms, nevertheless, EC2 API and GCE API projects are developing within OpenStack, aimed at developing interfaces that are compatible with EC2 and Google Compute Engine, respectively.

OpenStack supports deployment on top of a cluster running xCat, and is also developing its own Ironic project for the direct installation of the "cloud" on computer equipment.

In more detail the structure, principles and practice of working with OpenStack are described in.

Apache CloudStack is a software platform designed to create an IaaS cloud infrastructure and to automate the deployment, configuration and maintenance of a private, hybrid or public cloud infrastructure.

CloudStack does not depend on the type of hypervisor and allows you to use Xen (XenServer and Xen Cloud Platform), KVM, Oracle VM (VirtualBox) and VMware in a single cloud infrastructure (this platform property is called "hypervisor agnostic"). CloudStack allows you to organize the operation of both a public IaaS service, similar to Amazon EC2, and a private cloud infrastructure deployed on local servers and serving the needs of a particular enterprise. In the simplest case, a cloud infrastructure based on CloudStack consists of one control server and a set of compute nodes on which the guest OS is run in virtualization mode [8].

The main features of the platform:

– API support for major cloud platforms, in particular EC2, Citrix Cloud Center (C3) and vCloud API, as well as the OCCI standard;

– support for complete isolation of computing, network and disk resources;

– support for automatic allocation and resource limitation;

– availability of tools for generating reports and monitoring in real time;

– web interface based on the use of AJAX technology;

– tools that provide visual infrastructure management and daily tasks;

– the possibility of organizing a service that provides the lease of computing resources;

– support for network virtualization by isolating network segments into separate VLAN;

– provision of computing resources on demand, depending on the load created by the virtual environment;

– full automation of the allocation of storage space, computing and network resources for the entire physical infrastructure, including the ability to define resource allocation policies and support load balancing;

– availability of tools for managing the creation of "snapshots" of environments and backups;

– failover tools that support automatic recovery of virtual machines after the failure of the server on which they were running.

CloudStack has flexible scalability, supports the deployment of infrastructures serving thousands of hosts, and allows you to manage cloud systems that span multiple geographically separated data centers.

OpenNebula is an open and extensible IaaS platform that allows you to deploy a private, public, or hybrid cloud service on existing servers, similar in functionality to Amazon EC2. Different hypervisors (Xen, KVM, VMware ESX / ESXi) can be used on the physical server and cluster at the same time. Any of the systems supported by these hypervisors can work as "guest" OS.

The modular architecture allows OpenNebula to integrate with any virtualization platform, data warehouse or management manager. All inherent cloud technologies and functions are supported, including Live Migration (a virtual machine can be easily migrated to another server) [9].

OpenNebula implements its own API and also supports EC2, OCCI and vCloud.

The open platform Nimbus is designed to create IaaS-level cloud services compatible with EC2 and S3, and is focused on use in scientific communities.

Nimbus supports the Xen and KVM hypervisors as well as the Portable Batch System and Oracle Grid Engine virtual machine schedulers, providing all the core functionality found in cloud IaaS platforms. The system source code is distributed under the Apache License, version 2.

OpenQRM is free and open source software for managing data center infrastructure and building private, public and hybrid IaaS cloud systems.

The platform developers declare their adherence to the principle of strict separation of hardware (physical servers and virtual machines) and software (OS images). In this case, any equipment is understood as a computing resource that must be replaced without the need to reconfigure the software.

Features of OpenQRM:

– the ability to integrate with all major technologies for creating data warehouses (both open and commercial);

– support for Windows, Linux, OpenSolaris and FreeBSD;

– support for a large number of hypervisors and containers: KVM, Xen, Citrix XenServer, VMWare ESX / ESXi, lxc, OpenVZ and VirtualBox;

– support for configuring hybrid "clouds";

– supports popular open source cloud service management tools Puppet, Nagios / Icinga and collectd;

– over 50 plugins that extend the capabilities of the system and support integration with the user's infrastructure [9].

While IaaS provides the user with a "bare" virtual computer that has yet to be configured, PaaS provides access to a "computer" with an already installed and configured OS, system programs, programming languages, development tools, DBMS, etc.

The emergence of PaaS platforms has become a natural stage in the development of cloud infrastructures aimed at solving typical tasks, for example, providing web hosting services.

Cloud Foundry is an open PaaS platform that allows you to create an infrastructure for running in cloud environments of final applications in Java (with support for the Spring framework), Groovy (Grails), Ruby (Rails, Sinatra), JavaScript (Node.js) , Scala, and other languages running in the Java Virtual Machine. The platform makes it possible to work in conjunction with container virtualization based on Docker and supports operation on top of deployed IaaS platforms VMware vSphere, Amazon Web Services and OpenStack, as well as deployment on local systems. From a DBMS, Cloud Foundry allows you to work with MySQL, Redis and MongoDB. Cloud Foundry was developed by VMware. The platform code was opened in 2011 and has since been distributed under the Apache License 2.0. The platform is currently being developed by the Cloud Foundry, whose key founders are EMC, HP, IBM, Intel, Pivotal, SAP and

VMware. There is also a version of Pivotal Cloud Foundry developed by Pivotal, as well as the Cloud Foundry codebased platform HPE Helion Stackato developed by Hewlett-Packard [10].

OpenShift is a cloud-based PaaS platform developed by Red Hat for web application development and hosting. Red Hat currently supports several flavors of this platform. OpenShift Origin is open source and licensed under the Apache License 2.0. OpenShift is designed to install programs that run on Red Hat Enterprise Linux. Major programming languages: JavaScript, Ruby, Python, PHP, Perl, Java, Haskell and .NET. Among DBMSs, OpenShift supports MySQL, PostgreSQL, MongoDB and SQLite. The platform supports a number of popular web application development frameworks: Rack (Ruby), WSGI (Python), PSGI (Perl), and Node.js (JavaScript). In addition to them, the basic set of applications also includes frameworks Laravel, CodeIgniter, CakePHP, Ruby on Rails, Django, Perl Dancer, Flask, Sinatra, Tornado and Web2py.

The OpenShift platform uses the concept of "cartridges" - plug-ins that serve to expand the capabilities of the system. Using the OpenShift Cartridge API, developers can add support for additional programming languages, DBMS and middleware components to the system. In addition, Red Hat maintains application repositories that allow the user to download the languages, development tools, DBMS, and more that they need.

The Cloudify platform is based on TOSCA (Topology and Orchestration Specifications for Cloud Applications), a standard and language for describing cloud services developed by the OASIS Foundation [11].

The goal of the TOSCA standard is to enhance the portability of cloud applications. According to the developers, TOSCA will allow describing the interaction of applications and cloud service infrastructure, the relationship between service components and the behavior of the services themselves, regardless of the service provider or developer of a particular cloud technology.

The Cloudify platform can be deployed on top of OpenStack, AWS, CloudStack, Microsoft Azure and VMware.

Apache Stratos is an extensible PaaS framework that provides the means to run Apache Tomcat, PHP, and MySQL applications in dedicated environments running on top of common cloud infrastructures. Stratos provides a ready-to-use cloud environment for developing, testing and running scalable applications, taking on tasks such as automatic resource management, load balancing, monitoring and billing.

Among the main features of Stratos:

– ease of deployment of PaaS infrastructure (test configuration can be run on a developer's computer);

– Availability of support for multi-tenant PaaS hosting applications similar to Google App Engine. The platform provides tools for monitoring resource consumption and coordinating the cancellation of funds available on the personal account;

– simplified methods of forming cartridges (modules) for launching new types of applications. Stratos can be quickly adapted to support new programming languages, operating systems and DBMSs;

– a neutral programming model that does not require reworking programs running in the PaaS;

– automatic scaling with increasing resource demand. Using the Complex Event Processor (CEP) to make real-time resource reallocation decisions based on rules and system state;

– the ability to distribute infrastructure across various cloud platforms. For example, if there is a lack of resources in the local private cloud, you can use the power of public cloud systems;

– support for working on top of various IaaS environments created on the basis of OpenStack, CloudStack, Amazone ES2 SUSECloud and VMWare vCloud.In theory, Stratos can be used with any system for which the Apache jclouds API is available;

– online disaster recovery and high availability tools;

– availability of a web interface and a command line interface for infrastructure administrators and users. A REST API is provided for integration with other cloud systems [11].

1.3. Analysis of advantages, disadvantages, risks and problems of cloud computing usage

From the supplier's point of view, due to the pooling of resources and the volatile nature of consumer consumption, cloud computing saves scale by using less hardware resources than would be required for dedicated hardware capacity for each consumer, and by automating resource allocation modification procedures. significantly reduces the cost of customer service.

From the consumer's point of view, these characteristics allow to obtain services with a high level of availability (English High availability) and low risks of disability, to ensure rapid scaling of the computer system due to flexibility without the need to create, maintain and upgrade its own hardware infrastructure.

Convenience and versatility of access is provided by wide availability of services and support of various class of terminal devices (personal computers, mobile phones, Internet tablets). Thus, the goal of our note, as always good, is to systematize the basic information related to this topic and put everything on the shelves. The use of cloud computing, along with numerous benefits, entails some risks associated primarily with the user's dependence on the cloud provider. This dependence is much stronger and has more aspects than dependence on traditional software vendors. One aspect of the problem is the binding of the user to a specific vendor (the so-called vendor-lock). If the provider begins to dictate unacceptable conditions to users, the latter will be forced to either accept these conditions or stop using the service. If the supplier for some reason leaves the market, then the service provided by him may disappear along with him. When using traditional software, these problems did not arise: a legally acquired copy of the program can be used even after its manufacturer changes its conditions or ceases to exist. One way to combat this problem is to standardize cloud services. For software developers working in the "clouds", the most important thing is the presence of a standard interface (API) for working with cloud services. A number of successes have been achieved here over the past few years. On the one hand, there is a de facto standard - EC2, on the other hand, the open standard OCCI (Open Cloud Computing Interface) has been developed and is gaining more and more popularity. At the same time, cloud platforms that support both standards are increasingly appearing. Another way is to isolate the user software from the rest of the system, in particular, using container virtualization. So, using the Docker package, you can run processes in an isolated environment - a kind of "sandbox", where besides the process itself, only its descendant processes exist. Although the process runs in the same OS as other, ordinary processes, it simply does not see them. Thus, using Docker, a developer can separate his application from the system, place it in a Docker container and, if necessary, transfer it to another system of the same type.

The next group of questions concerns data security. Who owns the data center? Under whose control? Is the data center located inside or outside the jurisdiction of the cloud service owner? A widespread situation is when a supplier company uses data centers located in different countries. At the same time, the state on whose territory the data center is located can access any information that is stored in it. For example, according to the laws of the United States, where the largest number of data centers are located, in this case the supplier company does not even have the right to disclose the fact of transferring confidential information to anyone other than its lawyers.

In the case of cloud services, users' personal data is stored on remote servers, which requires a higher level of trust in the provider. In addition, cloud services run on computers that are not controlled by the user. This limits the ability to study the program in operation and reverse engineering to ensure compatibility, which is enshrined in Russian legislation. In this regard, logical solutions are either self-deployment of cloud infrastructure (IaaS), or the use of an existing cloud platform (PaaS).

The main advantages of cloud computing:

– The user pays only when he needs the service and exactly for what he uses. This flexible pricing scheme allows you to significantly reduce costs.

– No costs for purchasing, maintaining and upgrading software and hardware.

– Scalability, fault tolerance, virtualization and security - automatic provision and release of the necessary resources depending on the number of users served by the application. The service of the provided resources falls on the shoulders of the provider.

Updates of all provided software resources occur on the cloud side, more regularly and in a timely manner.

– The ability to create a document or program and share it within the development team dramatically improves application development productivity. Ability to define, change and track execution schedules, tasks, areas of responsibility, roles (designers, developers, testers, QA based on access rights).

– Remote access to cloud computing - in fact, you can work with cloud computing from anywhere on the planet where there is Internet. To work with the cloud, usually no specialized software is required, a browser is enough.

Currently, we can talk about the existence of two types of cloud technologies, which differ in the nature of interaction with equipment: technology based on virtualization, and technology that involves the direct installation and operation of the "cloud" on the computer "hardware". Virtualization technologies have evolved since the mid-1960s. At that time, the OS was called the supervisor. Subsequently, when it became possible to run one OS on top of another, the term hypervisor appeared, denoting a program (or, less often, a device) that allows you to run several OSes simultaneously on one computer. The hypervisor manages resources and shares them between different operating systems, isolates running operating systems from each other, and also ensures their interaction (file exchange, network interaction, etc.). The hypervisor itself is in some way a minimal operating system. It provides a virtual machine service running under its control. In doing so, it emulates the real (physical) hardware of a specific machine and controls these machines, allocating and freeing resources for them. The hypervisor allows independent "power on", reboot, "shutdown" of any of the virtual machines with a particular OS. From the point of view of the guest (running a hypervisor) OS, a virtual system is no different from a physical one.

There are three main types of software hypervisors: standalone, guest, and hybrid. A stand-alone hypervisor (VMware ESXi, Xen) is capable of running on "bare" hardware, that is, it does not require an OS and directly provides operating systems with access to hardware. This type of hypervisor provides the best performance and therefore is used to work with server operating systems. The guest hypervisor runs on some underlying OS and performs all I / O through a user-level process running under this system. The hybrid hypervisor operates autonomously and contains a special service OS through which the guest systems gain access to the hardware [12].

In addition to virtualization using hypervisors, there is the so-called container virtualization. The difference between these approaches is as follows. The hypervisor emulates the hardware on top of which the guest OSs run. At the same time, everything that the equipment "knows" to do must be available to the guest OS from the base OS (ie, the "host" machine). In contrast, containers are virtualization at the operating system level, not at the hardware level: each guest OS uses the same kernel (and in some cases other parts of the OS) as the base one. As a result, containers are smaller and more compact than hypervisor guest environments because they have much more in common with the base.

For software developers who intend to create and use their applications in the cloud, the MaaS, IaaS and PaaS tiers are of greatest interest. The cloud is built on one or several servers connected by virtualization systems. Also, virtualization technologies allow you to divide the hardware capacity into parts that meet the current needs of users who turn to hardware as a service. As a result, the user moves from purchasing, managing and amortizing hardware resources to purchasing server time, disk space, and network bandwidth needed to complete their tasks. Container virtualization also has its limitations. So, due to the shared use of the kernel, different types of guest operating systems cannot be run on the same server. For example, it is not possible to run FreeBSD or Windows on a Linux container system, although you can run various Linux distributions. For hypervisors, this problem does not exist. The most famous open-source container virtualization software packages are OpenVZ, LXC and Docker, which is an add-on over LXC [12].

Note that although virtualization technologies play an important role in organizing cloud computing, the NIST list of inherent properties of these computing does not contain the concept of virtualization.

Now let’s consider the main disadvantages. Firstly, the cloud provides weaker legal protection - data in the cloud is less protected in the event of any action taken by law enforcement and other structures. Government agencies or investigative attorneys may well be able to obtain such data without a search warrant. Secondly, need to transfer business data to a third party service provider. Vulnerability scanning or rigorous testing requires explicit authorization from the cloud service provider. Otherwise, it will be tantamount to the fact that the client is trying to gain unauthorized access to the provider's systems. Strict operating rules and user training are required. Always ready to serve from anywhere that phishing attacks that employees at home are exposed to could threaten the company they work for.

Conclusions on the First Part

Existing types of cloud services, hardware, software and organizational features of their implementation are considered. A comparative analysis of the advantages and disadvantages of existing technologies for creating cloud services is carried out. The problems and risks that are relevant from the point of view of cloud service users are highlighted. An analytical review of free cloud hardware and software platforms has been performed. The results of the review can be useful if it is necessary to solve the problem of choosing a free cloud hardware and software platform depending on the requirements for the developed software systems that involve the implementation of a cloud service.

We can point out some advantages of using cloud technologies:

- Information is available from all types of devices that are connected to the Internet.

- Reducing the cost of purchasing expensive powerful computers, servers, there is no need to pay for the work only on the elements you used.

- The necessary tools to work are automatically provided by the web service.

- High manufacturability of computing power allows you to store, analyze and process data.

- Services are paid only if necessary, payment is made only for the required package of services.

- Modern cloud computing can provide the highest reliability, in addition, only a small number of organizations can afford to support a full-fledged data center.

Disadvantages of cloud technologies:

- To work with the "cloud" requires a constant connection to the Internet.

- The user may not always be able to customize the software used for personal use.

- To create your own "cloud" will require very high costs, which is impractical for new businesses.

- "Cloud" is a data repository that can be accessed by attackers through system vulnerabilities.

Cloud computing is a combination of several key technologies that have been developed over the years and are considered by many researchers as the next generation.

Companies that create a system of infrastructure in the cloud on their projects will only benefit, as you only need to pay for what you use. If you use these instructions correctly, the whole room can be configured with the help of one DevOps engineer. We use the EC2 instance only to the capacity required by the project, That is why companies save a lot of money.

PART 2
SELECTIONS OF CLOUD PLATFORMS AND SERVICES FOR CREATING CONTINIOUNS DEPLOY OF DEVELOPMENT TEAM AND CONFIGURING SECURITY

Comparison of network function is an important aspect, since creating an isolated cloud requires not only a source of computing resources, but also a separate VPN access and network address.

In Amazon Web Services, you can use Virtual Private Cloud to create a VPN with subnet, routing table, private IP ranges, and network gateways. In addition, there is Route 53 service to implement DNS web service.

Microsoft Azure also offers extensive networking tools. Virtual networking (VNET) allows you to set up a VPN, set up a public IP, connect to a hybrid cloud, and enable firewall and DNS [13].

Google Cloud Platform's offerings are not as extensive. The platform so far only has a Cloud virtual network with support for Public IP subnets, its own firewall, and the necessary DNS settings.

These market players are closely related when it comes to their core features and services. They manage the main elements of the public cloud:

self-service;

instant provision;

autoscaling;

safety;

conformity;

features of identity management.

2.1. Analysing the cloud computing bases of leading hyperscalers in the market nowadays

Cloud technologies have firmly entered the daily life of users, and for large and medium-sized businesses they have become an integral part of information and technological processes.

Companies that fit the definition of hyperscalers account for almost 70% of the cloud services market. In 2018, hyperscalers accounted for less than half (47%) of the cloud services market, which shown in Fig.2.1.

The largest hyperscalers in the world are Amazon Web Services, Google, Microsoft and IBM. Recently, the Chinese company Alibaba Cloud has been breaking into this market. 45% of hyperscaler data centers are located in the US and 8% in China.

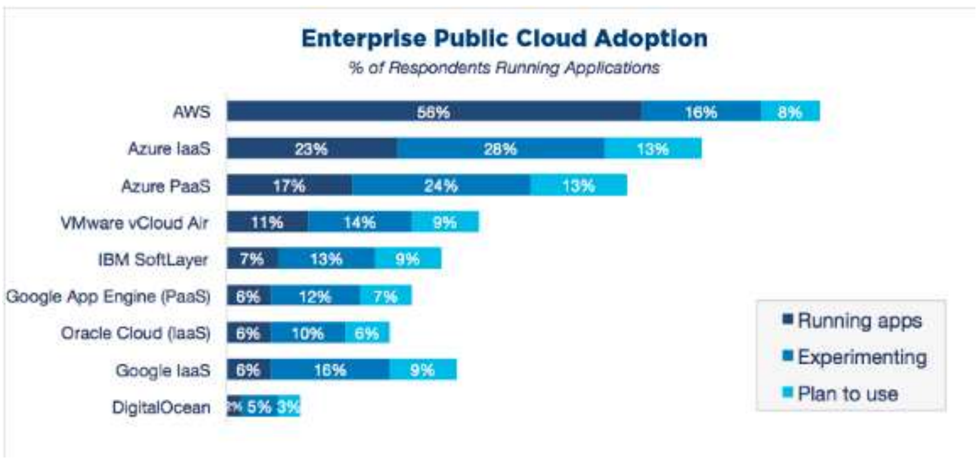


Fig. 2.1. Enterprise Public Cloud Adoption

At the beginning of 2019, the market shares of the major hyperscalers were as follows. The absolute market leader, AWS, holds about 34% of the market. The other three leaders together (MS, Google, IBM) hold about 27%. All the rest - no more than 10%, however, their share tends to grow (Fig.2.2).

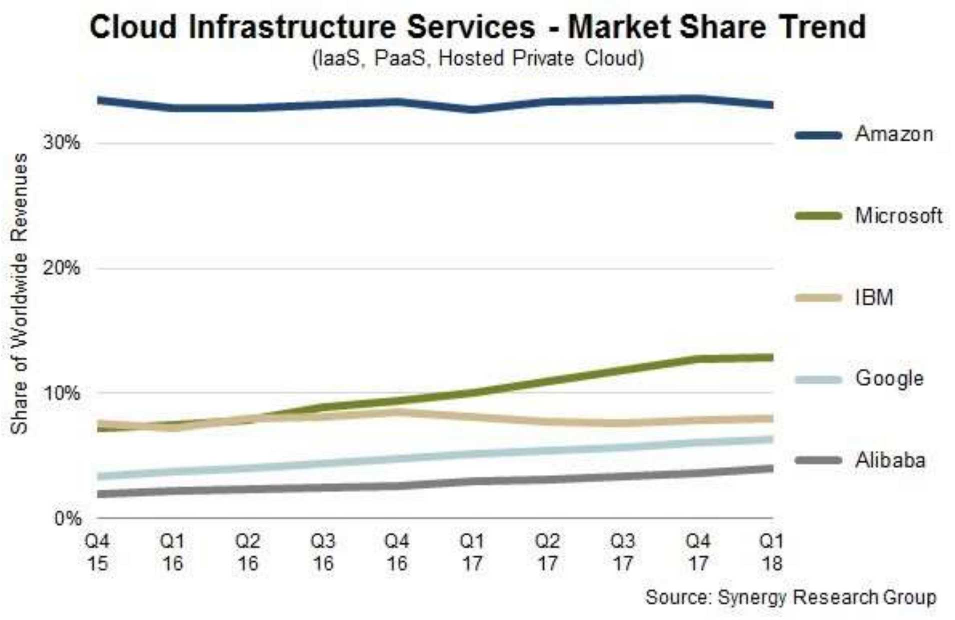


Fig. 2.2. Market shares of major hyperscalers

However, in Ukraine, the picture of the multi-cloud services market may be radically different. The fact is that AWS cloud services in our country are not represented to the same extent as in the rest of the world, for a very simple reason - the nearest AWS data center is located in Frankfurt am Main. Therefore, a bunch of problems immediately arise related to the regulation of the storage of personal data. For the same reason, less than in the rest of the world, Microsoft is represented in our country (MS Azure services are provided in Russia only through partner data centers).

Amazon Web Services (AWS), a division of Amazon, has been in the multi-cloud IaaS market since 2012. AWS offers integrated IaaS + PaaS services. The most popular service EC2 (Elastic Compute Cloud) provides customers with virtual machines with per second billing, both in general and individual use. AWS data centers are located in the United States, Canada, France, Germany, Australia, India, Japan, Singapore, South Korea, and Brazil. Distributed data centers of two AWS partner companies operate in China. AWS is a Tier 1 Internet Service Provider and has topped the multicloud rating for 10 consecutive years. AWS customer companies include companies that pay $ 100 million per year for AWS multi-cloud services. The company has more than 2,000 professional service partners worldwide and has the world's largest network of third-party software developers, which eliminates software licensing issues for multi-cloud customers [14].

However, for the most effective use of AWS services, a sufficiently high expertise of the customer's IT staff is required, despite the presence of consulting partners.

Google has been providing PaaS platform services since 2008. At the end of 2012, the Google Compute Engine project was launched. The Google Cloud Platform (GCP) multi-cloud platform is an integrated IaaS + PaaS platform, whose services include object storage systems, Docker container services (Google Kubernetes Engine), as well as relatively new features of the so-called. "Serverless computing" in beta testing. GCP virtual machines operate in per-second charging mode. Often multicloud services are provided through partners (Google Cloud Interconnect). The regional distribution of data centers has three-zone redundancy, as opposed to two-zone AWS. For example, the North American region has three zones: the west coast, the east coast, and the central states. Other Google data centers are located in Belgium, Germany, Holland, United Kingdom, Japan, Singapore, India, Australia, Brazil, Canada, and Taiwan. Hotline support is available in English and Japanese.

The portals are available in English, Dutch, French, German, Italian, Polish, Spanish, Turkish, Russian, Portuguese, Korean, Japanese, Chinese (in two versions) and Thai. Documentation is only available in English, German, Japanese, and Brazilian Portuguese. The most common applications that work effectively in GCP are big data and other analytics, machine learning, cloud applications, or other cloud-optimized applications. Google bills itself as an “open provider,” with an emphasis on portability for open ecosystems. Google automates cloud operations and does not use any “proprietary” functionality. GCP multi-cloud has a well-designed, reliable and efficient functionality for basic IaaS and PaaS services, although its range of services is not as wide as that of other market leaders.

A distinctive feature of GCP is an emphasis on analytics and machine learning, based on a cloud database with a high processing speed of huge amounts of Google BigQuery data. Google helps customers transform operational business processes through a dedicated Customer Reliability Engineering program, which is currently only available to a limited number of customers. The goal of the program is to educate customers on how to conduct operations in the cloud, similar to how Google engineers themselves do [15].

The multi-cloud platform "Microsoft Windows Azure" was launched on February 1, 2010. At first, the platform was represented only by cloud services, Azure SQL Database and Block & Table Storages data storage service. Now MS Azure has more than 40 data centers around the world and provides more than 120 services. We can say that this is the next big platform after .NET, which can be considered the "progenitor" of MS Azure. Geographic replication exists for the Azure SQL Database service. It is available by default for any service version and location, and provides replication to four additional regions. MS Azure is positioned as an "enterprise-ready" platform, that is, as it does not require much effort when implemented in an enterprise by its own IT staff. However, according to customer reviews collected in the Gartner study, this platform is sometimes not as "enterprise-ready" as they expected, given the fact that MS has traditionally strong positions in the corporate market, where MS products are used everywhere. MS clients especially do not like the problems with the technical support of MS Azure services, documentation, training and the ISV ecosystem. However, Microsoft is actively working on fixing these issues. Nevertheless, a number of problems associated with the partner ecosystem of managed services (lack of experience of partners, poor interaction of partners with different MS departments) still remain on the agenda. Partner implementation of the Azure Fast Start program often falls short of user expectations [16].

IBM on the Gartner Magic Quadrant has fallen into the category of "visionaries", as work on multi-cloud services in this company, as they say, "in progress." IBM is implementing the Next-Generation Infrastructure project, which should increase the scale and efficiency of the provision of cloud services, but so far there is no information on the completion date of this project. Basically, IBM cloud services are represented by SoftLayer, acquired in 2013, and, according to Gartner, since then their nomenclature has not undergone major changes. These services are aimed mainly at small and medium businesses, however, they lack many IaaS features that are in demand in this market. IBM also frequently faces engineering challenges that impact the time to market for cloud services. A project like the Bluemix portal, a beta version of which was presented in 2016, was frozen and merged with the IBM Cloud platform in 2017.

For customers, this creates risks of uncertainty in roadmaps, which affects interactions with their partner ecosystem [16].

Alibaba also got into the “visionaries” square of Gartner, because it is quite an ambitious and fast growing cloud provider. However, it should be noted that the Chinese version of the Alibaba cloud platform website has much more functionality than their English website. The company, however, cannot be classified as an innovator, since it, like many leading Chinese IT brands, is only actively adopting the successful experience and approaches of competitors. However, this does not give them a reason to relax, since there are many examples when Chinese companies, starting with outright copying of technologies, became innovative leaders.

Oracle regularly proclaims its cloud is better than Amazon's. Nevertheless, Gartner classifies the Oracle cloud platform as a visionary, and not of the highest rank, since it offers only the minimum required set of IaaS functionality, multi-cloud storage and the ability to create global programmable networks. The company does not yet have an impressive track record of successful multi-cloud projects, therefore, customers need to take into account the risks of technical implementation, and the IT staff of a potential customer of Oracle cloud services must have good technical expertise. However, according to Gartner, Oracle has realistic forecasts, taking into account their late entry into the global cloud services market, and has a reasonable roadmap for the development of basic functionality that will be attractive for the intended use cases [16].

Companys for cloud infrastructure providers as a service, including the aforementioned hyperscalers: AWS, MS, Google, Alibaba Cloud, IBM and Oracle is shown in the Fig.2.3.

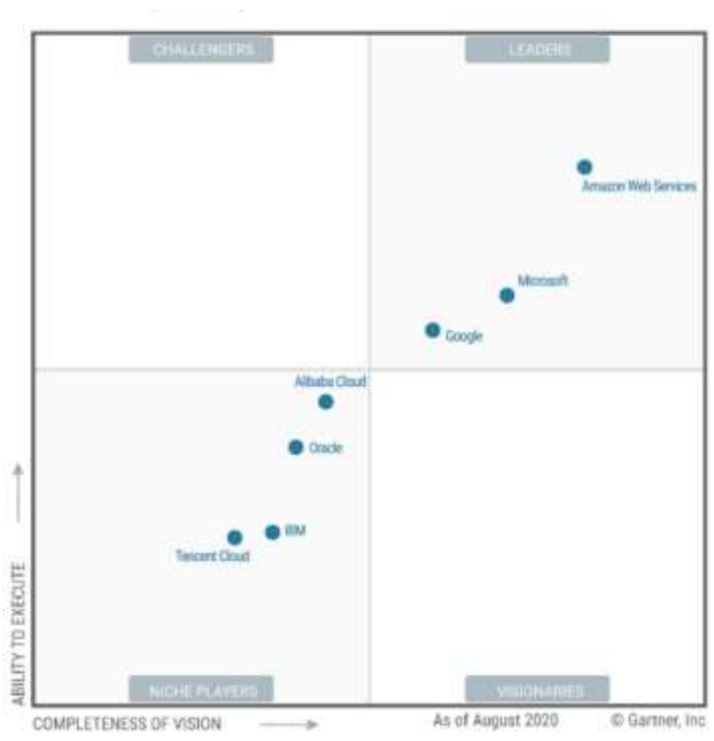


Fig. 2.3. Magic quadrant of cloud infrastructure providers as a service, including hyperscalers

Amazon Web Services. Amazon's cloud platform pioneered this area and has conquered a large market. With continuous innovations and improvements over the years, AWS has provided over 75 services with a wide range of coverage around the world. Servers are available in 14 geographic regions. The company's market share is growing steadily, with Amazon cloud technologies covering 34% of the market in the second quarter of 2020.Offers most of the infrastructure offerings such as low-level computing (EC2), storage (S3), VPC (networking), databases (RDS) with support for various operating systems (Windows, many flavors of Linux) with a large third party market. called the AWS Marketplace where vendors provide their add-ons. Pioneer for Serverless Computing with Lambda and now Fargate / Elastic Kubernetes Service. While Amazon offers more products and features, professional setup is required to operate and maintain. AWS gives you the building blocks, it's up to you to build it together. According to Marketing Intelligence for Cloud Service Providers currently, over 1 million customers make it one of the most popular computing platforms today: The price is similar to Microsoft [14].

Amazon Web Services features: convenient interface, rapid application deployment, well-documented services that make AWS easy to use, no power limits.

Microsoft Azure. The system was launched in 2010 and is developing at a very fast pace. Microsoft Azure is now a multifaceted complex system that supports many different services, programming languages, and frameworks. The cloud has over 60 services and data centers in 38 different geographic regions. Microsoft Azure currently holds 11% of the market. Supports Windows and Linux workloads with very deep integration into the Microsoft developer ecosystem with Visual Studio, .NET, and more. If you're an MSFT developer, Azure is an easy way to deploy your application. Price is similar to AWS

Microsoft Azure features: Ability to use any programming language, framework, or even tool, good scalability, full access to application connectors in Microsoft products.

Google Cloud Platform. Introduced in 2011, Google Cloud Platform is the youngest cloud platform and primarily meets the needs of Google and Youtube search.

The company currently has over 50 services and 6 global data centers. Google Cloud Platform has a 5% share of the cloud services market. The newcomer to the market mainly offers Platform as Service offerings such as Machine Learning as a Service, Kubernetes as a Service, etc. However, Google does not offer many product offerings. Google was a pioneer in Kubernetes and is a leader in delivering a truly managed Kubernetes experience Pricing may be cheaper than AWS and Microsoft, due to the PaaS nature of the product, less construction may be required.

Features of Google Cloud: good documentation, supports most regions: North America, South America, Europe, Asia and Australia, different storage classes, good prices.

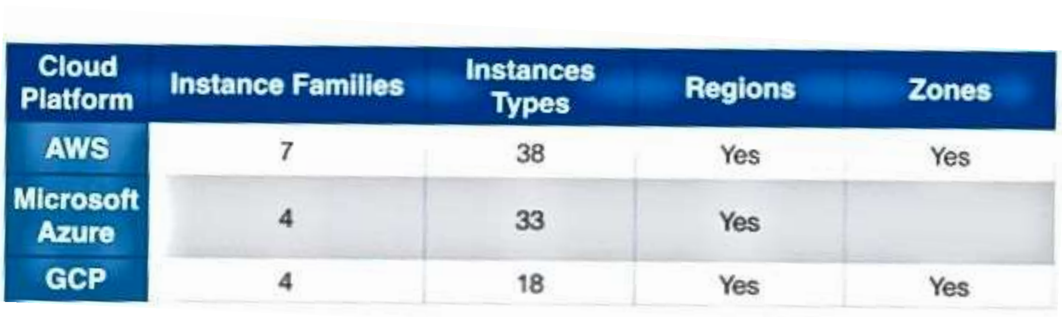
Now, let’s taok about main differences isTheir AWS, Azure and Google Cloud.Their API, Console and Identity Access Control. One of the benefits of cloud computing is automation. You can increase as well as decrease (like at night) and for that you automate this. But all services do it a little differently, so you can't take your knowledge from one provider and apply it to another. Also, the way you grant rights to users and (system) accounts is different. Google uses projects as a way to “decompose forces,” AWS doesn't have projects, but there are more general ways to share access, roles, and the like. Azure relies heavily on Active Directory. There are many other differences, such as prices, how virtualization works, and various services combined, but they came to mind first. Since these services are cheap, play around with all three, they have strong advantages. Google usually gives the best bang for your buck, but not in all areas. Azure is probably best suited if you already use a lot of Microsoft and Windows products. AWS is the most mature, has the most flexibility, the best console, but its VMs are less powerful or just more expensive IMO.

2.2. Choosing a platform, computer types, data storage and services for realization of continuous software development

Computing power is fundamental to the existence of an IT business. The advantage of cloud technology is that you always have a powerful and extensible tool at your fingertips with which you can interact remotely and scale at any time of the day, some ot this powerful tools shown in Fig.2.4.

Fig. 2.4. Comparison of the central computing service in different plstforms

In Amazon Web Services, the Elastic Compute Cloud (EC2) is the central computing service. EC2 has become a prime synonym for scalable on-demand computing. In order to plan even more carefully and reduce project launch costs, the company has introduced new sub-services such as AWS Elastic Beanstalk, Amazon EC2 Container Service. AWS currently supports 7 different instance families and 38 instance types (Fig.2.5.). It offers both regional support and zone support at the same time.

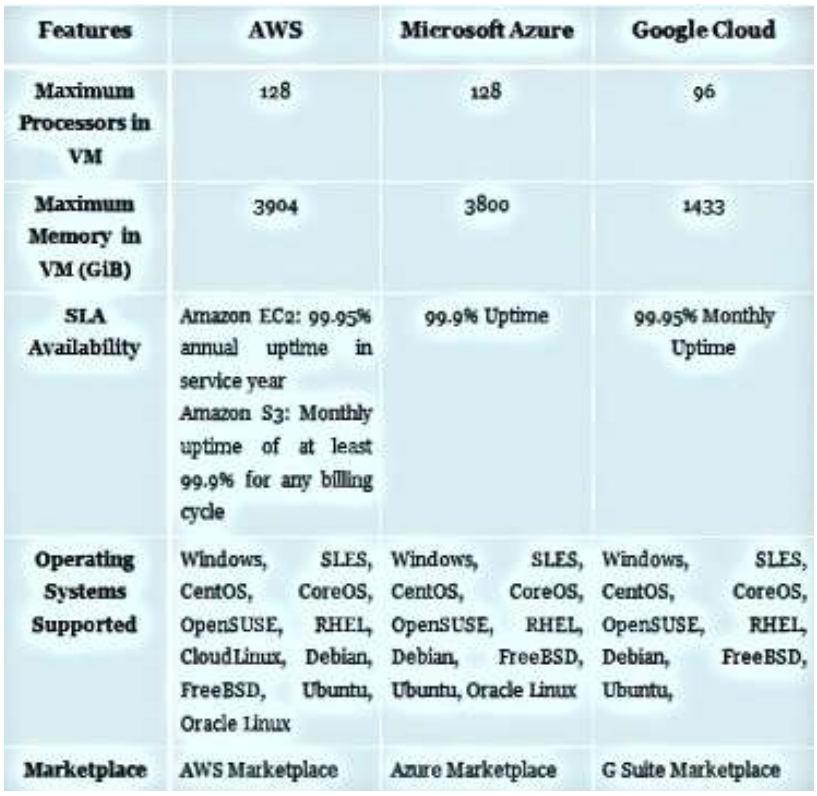


Cloud Platform	Instance Families	Instances Types	Regions	Zones
AWS	7	38	Yes	Yes
Microsoft Azure	4	33	Yes	
GCP	4	18	Yes	Yes

Fig. 2.5. Cloud Platform instance type presentation

The backbone of Microsoft Azure computing systems are classic virtual machines and high-performance Virtual Machine Scale Sets. Windows client applications can be deployed using a RemoteApp service. Azure Virtual Machine includes 4 different families, 33 types of instances that you can deploy in different regions. But support for a specific zone of the region is not yet supported.

Google Cloud Platform uses the Compute Engine service to handle computing processes. One of the main disadvantages is pricing, which is less flexible compared to AWS and Azure, presented on Fig.2.6.

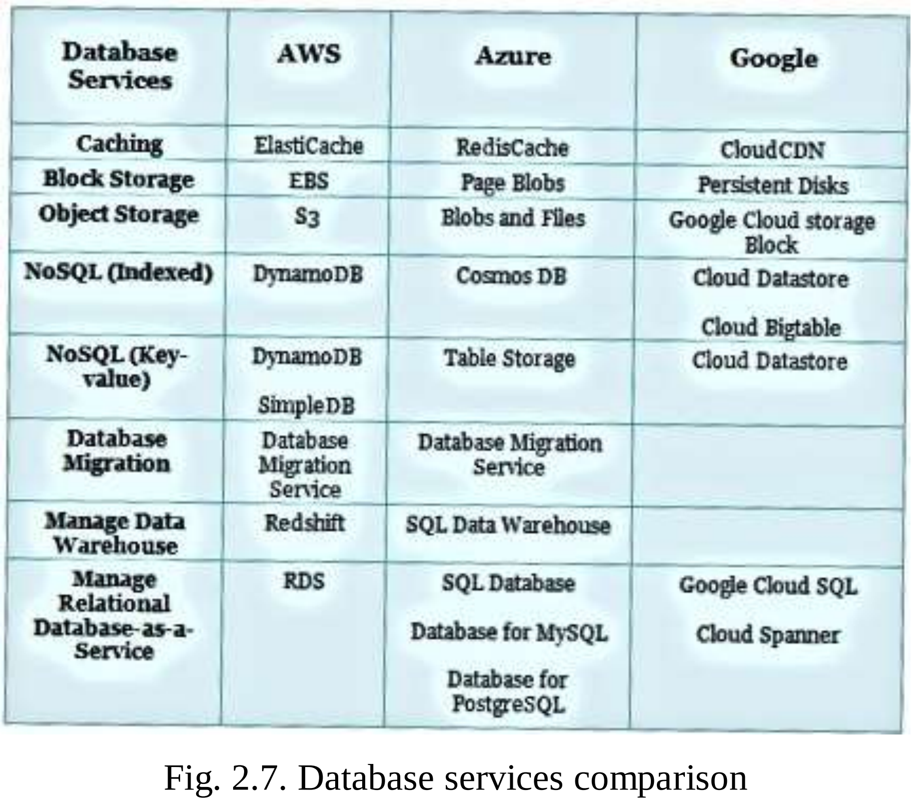


Features	AWS	Microsoft Azure	Google Cloud
Maximum Processors in VM	128	128	96
Maximum Memory in VM (GiB)	3904	3800	1433
SLA Availability	Amazon EC2: 99.95% annual uptime in service year Amazon S3: Monthly uptime of at least 99.9% for any billing cycle	99.9% Uptime	99.95% Monthly Uptime
Operating Systems Supported	Windows, SLES, CentOS, CoreOS, OpenSUSE, RHEL, CloudLinux, Debian, FreeBSD, Ubuntu, Oracle Linux	Windows, SLES, CentOS, CoreOS, OpenSUSE, RHEL, Debian, FreeBSD, Ubuntu, Oracle Linux	Windows, SLES, CentOS, CoreOS, OpenSUSE, RHEL, Debian, FreeBSD, Ubuntu,
Marketplace	AWS Marketplace	Azure Marketplace	G Suite Marketplace

Fig. 2.6. Comparison of Features and Solutions

Compute Engine supports most of the core cloud services - container deployment, scalability, and data processing. Google Cloud supports 4 instance families, 18 different instance types, and provides both regional placement and zone selection. If you choose the leader, then AWS and Microsoft Azure are now the most demanded cloud platforms. Computing capacities offered by companies are practically at equal levels, and the list of offered services is also constantly growing. Google Cloud Platform has also launched a separate direction for big data analytics and has great prospects for development in the future. Already, the Cloud Vision API, Cloud Speech API, and Google Translate API frameworks have multiple integrations in third-party services and applications.

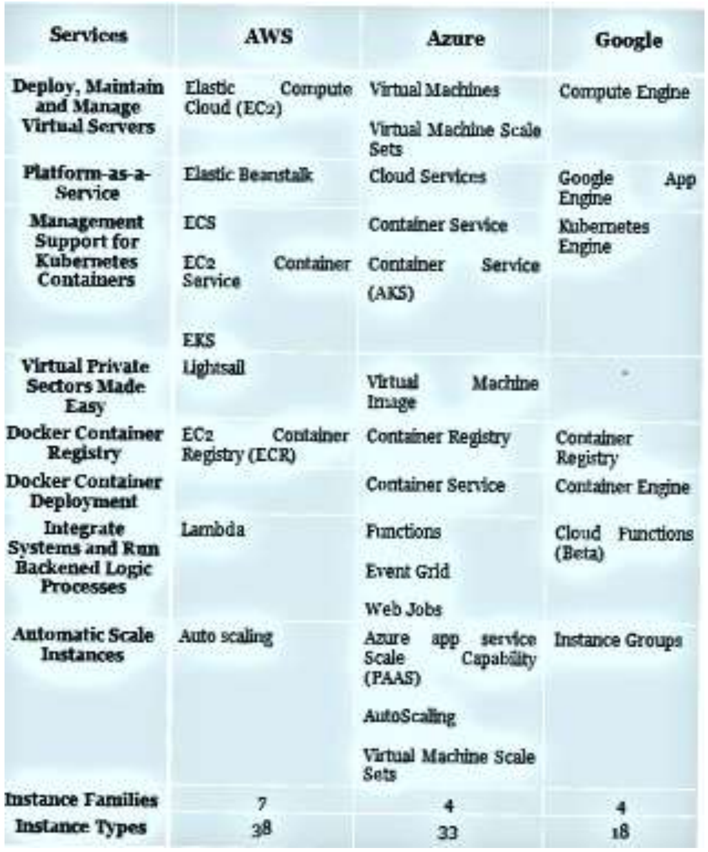
Information storage is the nexus of cloud computing because it allows all kinds of information to be collected in one repository. On Fig.2.7. we csan see, AWS Simple Storage Service, known as S3, is pretty much an industry standard. Overall, S3 created the notion of object-oriented storage, and a separate service, Amazon Glacier, was created to archive data. Azure and Google Cloud Platform also have fairly reliable and powerful storage facilities. The archive is sometimes referred to as "cold storage" because you won't be accessing the files it stores regularly. In this case, lower rates will bring you joy, but don't forget that speeds are lower too. Comparing the platforms, we can see that the archiving characteristics are similar, so your decision will likely depend on the API you have implemented in background [17].



Database Services	AWS	Azure	Google
Caching	ElastiCache	RedisCache	CloudCDN
Block Storage	EBS	Page Blobs	Persistent Disks
Object Storage	S3	Blobs and Files	Google Cloud storage Block
NoSQL (Indexed)	DynamoDB	Cosmos DB	Cloud Datastore Cloud Bigtable
NoSQL (Key-value)	DynamoDB SimpleDB	Table Storage	Cloud Datastore
Database Migration	Database Migration Service	Database Migration Service	
Manage Data Warehouse	Redshift	SQL Data Warehouse	
Manage Relational Database-as-a-Service	RDS	SQL Database Database for MySQL Database for PostgreSQL	Google Cloud SQL Cloud Spanner

Fig. 2.7. Database services comparison

Cloud services have quite different approaches to the pricing of using cloud services. AWS uses several payment models. On Demand: you only pay for the resources and services you use. Reservation: You choose the required number of resources that you want to order in advance from 1 to 3 years and pay on a usage basis. Often there are good discounts for such offers (up to 76%). Partial reservation. The more resources you use, the lower the cost of the services provided Rounding off in Amazon Web Services works based on hours of use. Microsoft Azure uses a more flexible pricing system, billing for the use of cloud resources, rounded to the minute. You can use out-of-the-box MSDN subscriptions with a specific amount of cloud money, or you can pay for resources in the usual way with monthly billing. Discounts also work based on the volume of services ordered. Google Cloud Platform has a similar billing system to Azure but rounds off resource usage over a period of 10 minutes. Each platform offers a pricing calculator to help you estimate costs. In terms of convenience, Microsoft Azure has come out ahead, the service offers to estimate costs not only using a calculator on the site, but also when creating new projects, directly in the management portal.



Services	AWS	Azure	Google
Deploy, Maintain and Manage Virtual Servers	Elastic Compute Cloud (EC2)	Virtual Machines Virtual Machine Scale Sets	Compute Engine
Platform-as-a-Service	Elastic Beanstalk	Cloud Services	Google App Engine
Management Support for Kubernetes Containers	ECS EC2 Container Service EKS	Container Service Container Service (AKS)	Kubernetes Engine
Virtual Private Sectors Made Easy	Lightsail	Virtual Machine Image	
Docker Container Registry	EC2 Container Registry (ECR)	Container Registry	Container Registry
Docker Container Deployment		Container Service	Container Engine
Integrate Systems and Run Backened Logic Processes	Lambda	Functions Event Grid Web Jobs	Cloud Functions (Beta)
Automatic Scale Instances	Auto scaling	Azure app service Scale Capability (PAAS) AutoScaling Virtual Machine Scale Sets	Instance Groups
Instance Families	7	4	4
Instance Types	38	33	18

Fig. 2.8. Services comparison of of high-tech platforms

2.3. Analysis of EC2 as a new way to software development

It would not be an exaggeration to say that Amazon radically changed the economic aspect of launching IT startups, it happened slowly and gradually, but now it is a fact. No one realizes how many companies are using Amazon EC2 anywhere in their infrastructure until it crashes and it looks like half the Internet is down. This does not mean that Amazon is just lucky, in fact, they have a very good product. Everyone uses this service because it greatly simplified the launch of applications and services, significantly reducing the amount of knowledge required, the steps that need to be taken and the money that is needed to launch a startup.

The first and foremost thing to know about EC2 is that it is not just shared hosting. It's best to think of it as hiring a part-time system and network administrator. Instead of hiring a highly paid employee to do the full work and automate everything for you, you pay a little more for each server, but you get rid of a whole host of problems. Power supply, network topology, hardware cost, incompatibility of equipment from different manufacturers, network storage devices - all these things had to think about back in 2004 (or get the idea out of my head). With AWS and its rapidly growing competitors, you don't have to think about such things until you want something more.

During the development phase, the ability to continuously test and deliver code allows you to increase the speed and quality of turnkey solutions. Continuous (several times a day) merging working copies of the program code into a common main branch and testing the results took shape in the "concept of continuous integration and software delivery".

It is important to understand that all methods and technologies are not mutually exclusive, but complementary. The only question is which methodology and technology will take a certain place in the overall IT ecosystem.

Today's business environment places significant demands on organizations and their ability to deliver software. Business leaders need flexibility. Developers need the ability to create new tools and features on a daily basis. Operations engineers need the ability to provision environments on demand. Customers want the latest features to be available and accessible across all platforms they use.

Segregation of stakeholder responsibilities. Participants in the development process and consumers of the finished project share the responsibility for one or another stage of the product life cycle. Developers and designers design business logic and also provide a positive user experience with the finished system. Quality engineers introduce end-to-end features and acceptance tests, DevOps engineers organize the logistics of the code, and users provide feedback on the results of using the system.

Risk reduction. Each group of participants in the development process minimizes possible risks when the product passes through the stages of the life cycle (optimization of data storage and processing, migration, etc.).

Short feedback loop. In commercial development, the speed of reaction to errors or requests for new functionality lays the foundation for the competitiveness of the future system. To add new functionality to the product faster than competitors, it is necessary to strive to automate the assembly and testing of code. However, in situations where a solution requires human participation, automation can only do harm. For such situations, it is recommended to reduce the number of information intermediaries, ensuring a short feedback loop.

Implementation of the environment. The development team needs a single working environment to control versions and build sub-branches for quality control, acceptability, scalability, and resiliency of the code produced. As control proceeds, the tested modules should be moved to the main branch of the project and sent for testing and assembly as part of a single solution. At the stage of final testing, the code is also evaluated from a security standpoint.

The method ensures the promptness of the output of new product functionality (working with customer requests). As a rule, these are a few days or weeks. At the same time, with the classical approach to developing client software, this can take a year.

In addition, the development team receives a pool of code alternatives, which optimizes the resource costs for solving the problem (by automating the initial testing of the functionality).

The quality of the product is improved by parallel testing of the functional blocks of the future system. Bottlenecks and critical points are captured and worked out early in the cycle.

However, project managers erroneously accept the methodology as a panacea and seek to incorporate it into all their designs. Lack of experience leads to the complication of work on the company's IT products.

The organization of interaction between project teams also deserves attention, since CI / CD is strongly tied to the human factor. Engineers, scrum specialists, analysts, dev teams, and other functional units must work in a unified ecosystem with adequate leadership and project management.

Writing code. Each of the developers writes the code for their module, conducts manual testing, and then merges the work result with the current version of the project in the main branch. For version control, the Git system is used, or similar solutions. When team members publish their module code to the main branch, the next phase begins.

Assembly. The version control system launches automatic building and testing of the project. Triggers for starting a build are configured by the team individually - committing changes to the main project branch, building on a schedule, on demand, etc. Jenkins or a similar product is used to automate the build.

Manual testing. When the CI system has successfully verified the performance of the test version, the code is sent to testers for manual inspection. In this case, the test assembly receives a candidate number for a further product release (for example, ver.1.1.1).

Release. As a result of manual testing, the assembly receives fixes, and the final candidate version number is increased (for example, after the first fix, the v.1.1.1 version becomes ver.1.1.2). After that, a version of the code to the client is released (for example, v.1.1.1) and the next stage of the cycle begins.

Deployment. At this stage, a working version of the product for clients is automatically published on the developer's production servers. After that, the client can interact with the program and get acquainted with its functionality both directly through the ready-made interface and through cloud services.

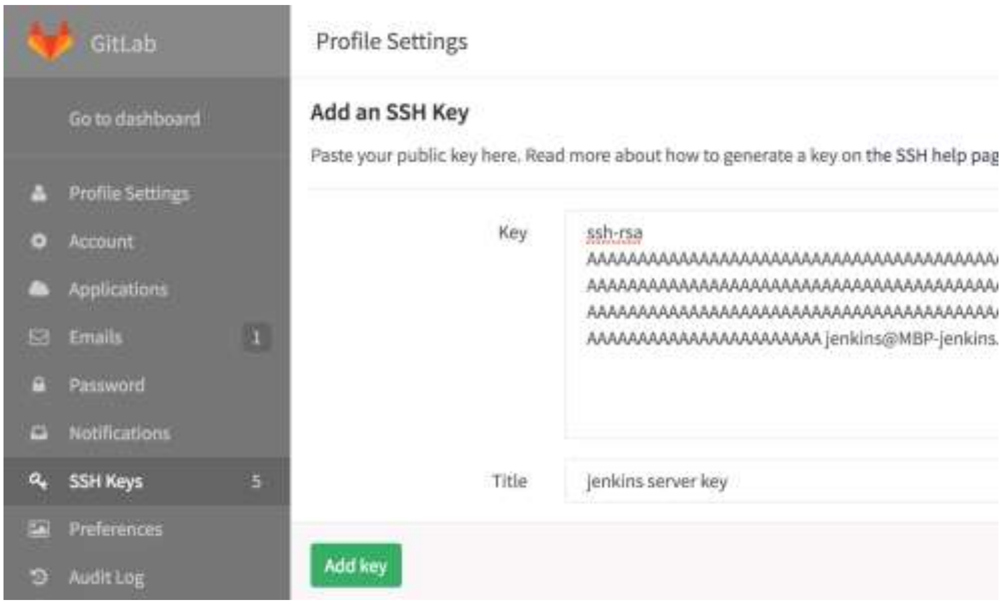
Support and monitoring. End users get started with the product. The development team supports it and analyzes the user experience.

Planning. Based on the user experience, requests for new functionality for the product are formed, and an improvement plan is being prepared. After that, the loop closes and goes to the initial stage - writing code. Next, a new iteration of CI / CD development begins.

2.4. Choosing a method for loading data into storage repositories and into EC instances

To configure security, we will use the Role-based Authorization Strategy plugin. Let's go to the settings through the Jenkins web interface, and then to the Manage and Assign Roles section. Here we can create various roles that are assigned to users and groups, and define their rights to certain operations. Here, everyone does everything at his own discretion. But, if you have configured external access to the server, I strongly recommend that you remove all rights for the "guest" user.

In the settings of GitLab itself, you must specify the corresponding public key (Fig.2.9):



Add an SSH Key

Paste your public key here. Read more about how to generate a key on the SSH help pag

Key

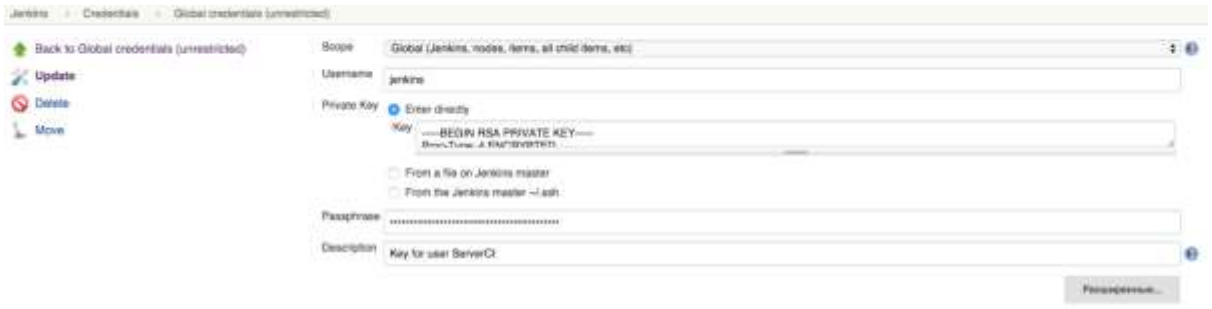
```
ssh-rsa
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAA jenkins@MBP-jenkins.
```

Title: jenkins server key

Fig. 2.9. Additing a ssh key into our code repositories

In case SCM is in the same network with Jenkins, it is better to post-merge a hook with SCM to run the task on Jenkins, instead of constantly polling with Jenkins SCM for changes.

It's the same as asking you if you want to eat every minute than if you just came to the kitchen and ate if you were hungry. We can see this representation in the Fig 2.10.


Jenkins › Credentials › Global credentials (unrestricted)

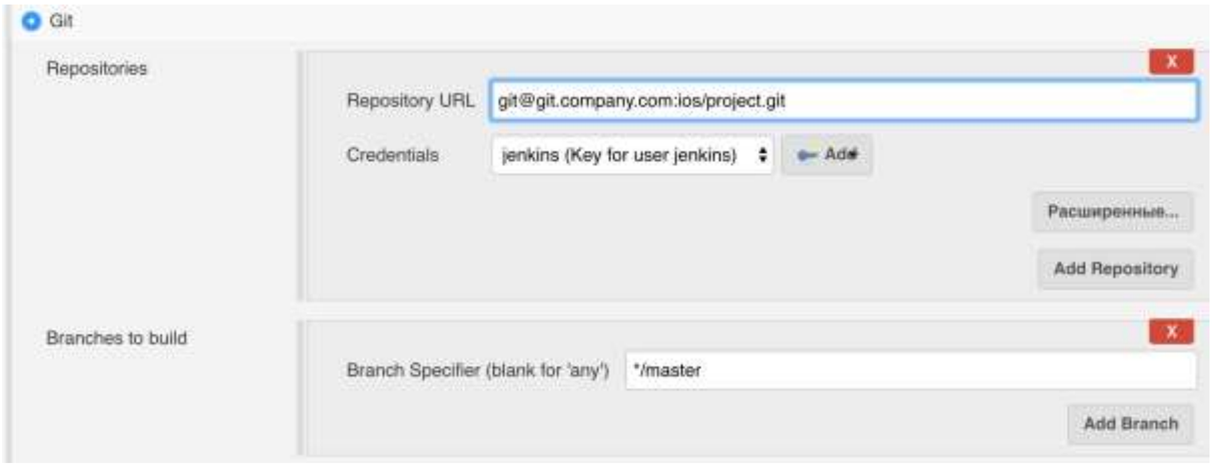
Back to Global credentials (unrestricted)
Update
Delete
Move

Scope: Global (Jenkins, nodes, items, all child items, etc)
Username: jenkins
Private Key: [✓ Enter directly]
Key: -----BEGIN RSA PRIVATE KEY-----
From a file on Jenkins master
From the Jenkins master ~/.ssh
Passphrase: ••
Description: Key for user ServerCI

Fig. 2.10. Registry RSA Private Key in jenkins

In Jenkins, you can change the working directory to whatever you like - and unload it along the desired path (but keep in mind that if there is more than one collector, you must a) bind the collector to the required machine b) bind the unload to this collector.

But if the github knows how to register external runners, then jankins is not needed at all. Register the runner on the machine where you want to download the sources and connect the pipe on the github (Fig.2.11):

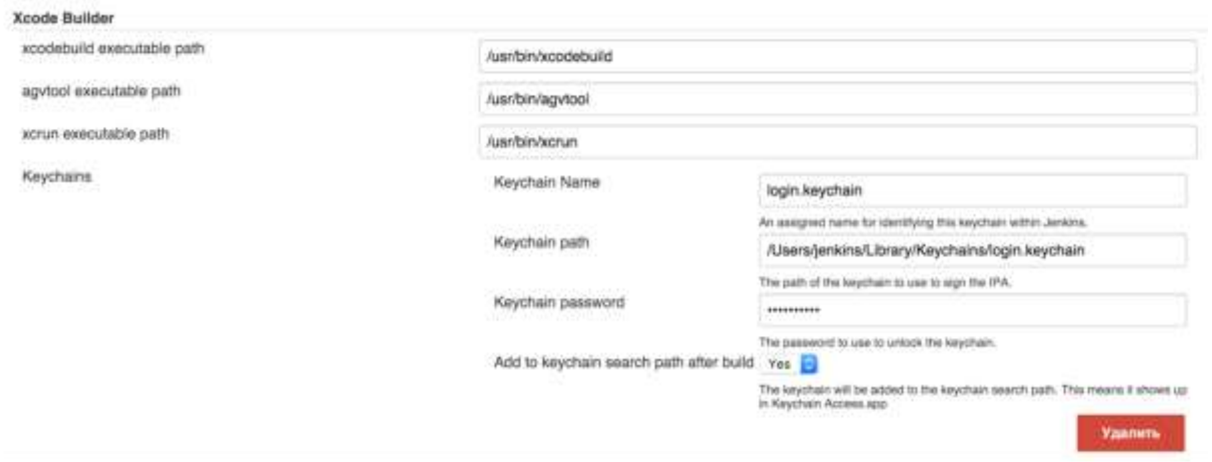

[✓ Git]

Repositories
Repository URL: git@git.company.com:ios/project.git
Credentials: jenkins (Key for user jenkins)
Add
Расширенные...
Add Repository

Branches to build
Branch Specifier (blank for 'any'): */master
Add Branch

Fig. 2.11. Realization of plugin for SCM

To deploy artifacts to Nexus, you will need to include the distributionManagement section in your pom. Nexus comes with specific repositories already configured for snapshots and releases. You have to provide the correct path to each one so that maven will deploy the snapshot and release the artifacts into the correct repo. Then every time you deploy artifacts - usually with mvn deploy or with the maven release plugin, the artifacts will be deployed there. Nexus has write authentication enabled by default,like on Fig.2.12. so you will need to make sure to add the server section with the correct credentials to the settings.xml of everyone who will deploy the artifacts. Jenkins can be viewed in much the same way as any other user. If you have to do this deploy as its build, then each build will be deployed to Nexus. There is also a post-build action for deploying artifacts if you want this to happen later in the Jenkins job.

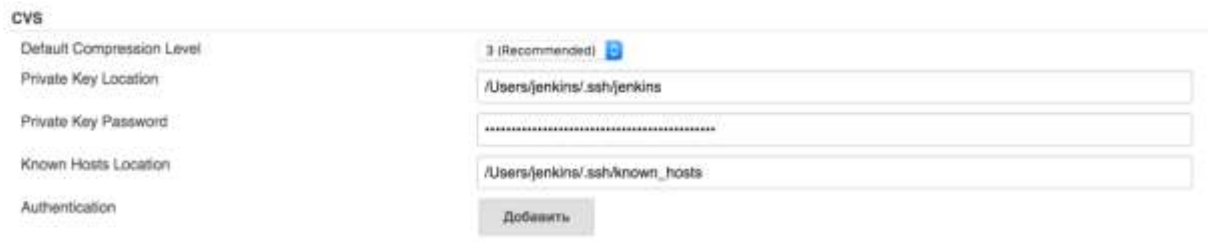


Xcode Builder

xcodebuild executable path	/usr/bin/xcodebuild	
agvtool executable path	/usr/bin/agvtool	
xcrun executable path	/usr/bin/xcrun	
Keychains	Keychain Name	login.keychain An assigned name for identifying this keychain within Jenkins.
	Keychain path	/Users/jenkins/Library/Keychains/login.keychain The path of the keychain to use to sign the IPA.
	Keychain password	********** The password to use to unlock the keychain.
	Add to keychain search path after build	Yes The keychain will be added to the keychain search path. This means it shows up in Keychain Access app

Fig. 2.12. Connecting instances with xCode

Now we can load the necessary certificates through the GUI or through the shell in login.keychain, and Xcode will automatically pull up the needed ones at build time. To configure access via ssh, do this: generate a shh-key, if it does not exist (instructions for an example can be found here) or add the key in the CVS section.



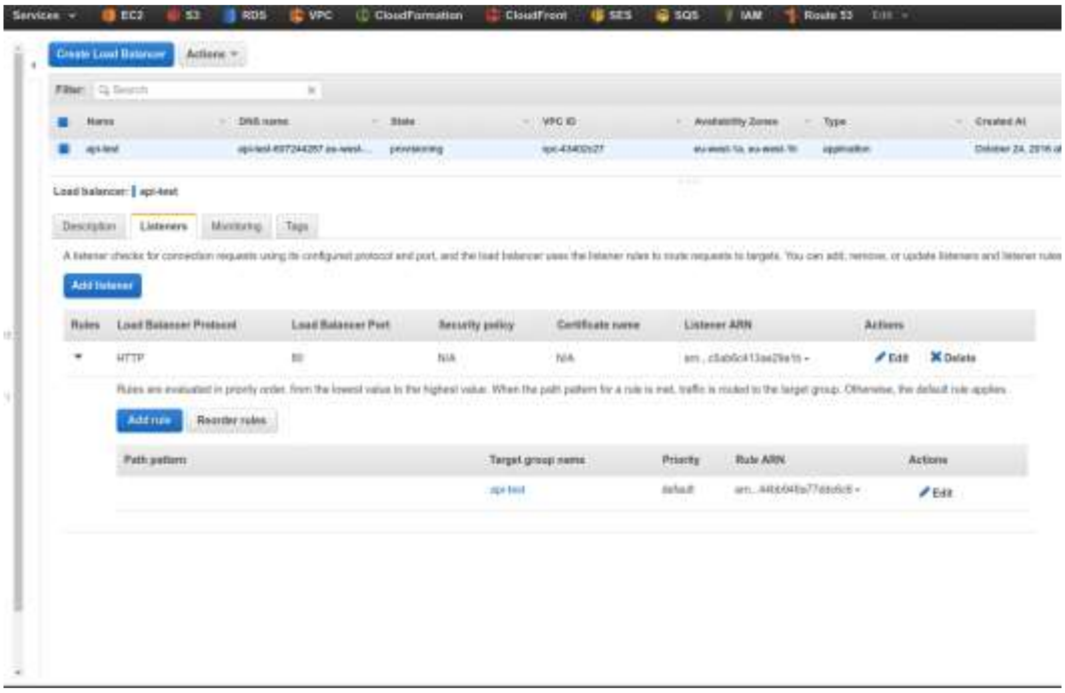
CVS

Default Compression Level	3 (Recommended)
Private Key Location	/Users/jenkins/.ssh/jenkins
Private Key Password	***
Known Hosts Location	/Users/jenkins/.ssh/known_hosts
Authentication	Добавить

Fig. 2.13. Manually set configuration for our keys

The next step is to set up Load Balancer. In today's web, and not just the web, we are forced to create applications that scale horizontally. Everyone has long known how such solutions are built. Most often, a bunch of servers are taken on which the application is deployed directly, moreover, it keeps all its state not locally, but in some DBMS. A load balancer is placed in front of these servers in the form of several more servers with Nginx or HAProxy. Users are directed to the balancer using DNS. But maintaining this entire economy is quite troublesome. And that's where Amazon comes to the rescue! Let's do a little experiment. You can repeat all the described steps yourself. Amazon won't charge you for this if you act quickly and clean up after yourself. Let's go to the AWS console, EC2 → Instances → Launch Instance. In the list of operating systems that appears, select Ubuntu Server LTS (HVM), SSD Volume Type, click Select. Let's take the type of intsection t2.micro, since it is for testing purposes. We will be asked about the keys with which we want to log into the EC2 instance. Select Create a new key pair, enter some name, click Download Key Pair. Save the resulting .pem file somewhere. Now click Launch Instances. You should see "Your instances are now launching". We are waiting for the instance to change its state in the list to green "running" [18].

Next, we need a load balancer. Enter the name test-load-balancer, click Continue. Select the HTTP protocol. Next, you need to configure the health check. Enter the address "/status.txt". At the Assign Security Groups step, add a check mark opposite web servers. At the Add EC2 Instances step, we do not add anything yet, and also uncheck the Enable Cross-Zone Load Balancing checkbox. In general, keeping servers in multiple AZs is a good idea, but in the context of this post, load balancing across multiple AZs will only complicate things. Please note that we skipped the step of specifying some tags there. As part of this note, they will not be useful to us, but in general, this is a very convenient thing with which an application through the AWS SDK can determine, for example, in which environment (dev / stage / prod) it works and not only. So, there is a balancer, but there are no cars behind it yet. Moreover, there is an image from which we want to create cars. To tie all this together, you need to create an Auto Scaling Group. Go to Auto Scaling → Auto Scaling Groups, click Create Auto Scaling Group, then Create launch configuration. Launch Configuration determines which machines we will start when the load increases. Choose My AMIs → nginx-image-v1 → Select. We select the type of instance t2.micro, it is desirable to use the same one from which the AMI was created. At the next step, enter the name launch-configuration-v1, then go to the Configure Security Group step, add permission to break into port 80 from anywhere. In fact, since these instances will be behind the load balancer, you might want to disable access to them directly over HTTP or SSH. But in general, it can be convenient in case of problems. When asked about the choice of keys, select the previously created key pair. In the step where you need to enter the name of the Launch Configuration, in the Advanced Details section, you may have noticed the User data field. This thing is similar to ELB tags, only it can be retrieved with a simple HTTP request without an access key and secret key, and also provides just data, not separated into keys and values. Again, this is not useful to us now, but in general the thing is very convenient. As a result we have complited load balancer in our system, as we can see in Fig. 2.14.



Name	DNS name	State	VPC ID	Availability Zones	Type	Created At
api-test	api-test-697244267.eu-west-...	provisioning	vpc-43403c27	eu-west-1a, eu-west-1b	application	October 24, 2016 at

Load balancer: api-test

A listener checks for connection requests using its configured protocol and port, and the load balancer uses the listener rules to route requests to targets. You can add, remove, or update listeners and listener rules

Rules	Load Balancer Protocol	Load Balancer Port	Security policy	Certificate name	Listener ARN	Actions
▾	HTTP	80	N/A	N/A	arn...c5ab5c413ae29e1b ▾	Edit Delete

Rules are evaluated in priority order, from the lowest value to the highest value. When the path pattern for a rule is met, traffic is routed to the target group. Otherwise, the default rule applies.

Path pattern	Target group name	Priority	Rule ARN	Actions
	api-test	default	arn...44bb04b1a77ddc6c8 ▾	Edit

Fig. 2.14. Load balancing config dashboard

Now that we've created the Launch Configuration, Amazon will finally allow us to create an Auto Scaling Group. We enter some name, let the initial number of instances be-2, select one of the subnets. In Advanced Details, select Receive traffic from Elastic Load Balancer (s), in the list we find the previously created balancer. At the next step, select "Use scaling policies to adjust the capacity of this group", we say to keep from 2 to 10 instances. Set up the conditions for resizing the group. I set to increase by 1 if the average CPU usage is over 70% and decrease by 1 if the CPU is used on average by less than 30%.

Conclusions on the Second Part

Cloud computing is a new technology that has been widely studied in recent years. Many cloud platforms have emerged from both vendors and academia. Dealing with all the issues and understanding how to use these platforms is an important task. A detailed comparison has been presented in this article, focusing on such aspects as architecture, characteristics, applications, etc.

From the analysis, users can better understand the characteristics and more reliably choose a cloud computing platform, according to protocols, interfaces, compatibility, implementation, deployment requirement, development capability, and so on. While each cloud computing platform has its merits, one thing is very important to emphasize - no matter which platform is chosen, there are still challenges.

Let us point out as an example the mechanisms of cluster failure in the cloud environment, guarantees of consistency and synchronization on different clusters in the cloud platform, standardization, security of the cloud platform and data in transit, etc. All these problems require solutions for each specific implementation. At the test site of the Faculty of Applied Mathematics and Control Processes of SP6GU, the cloud was built on a heterogeneous system, which consists of clusters of greatly different performance, built on two-, four-, and six-processor opterons.

Nevertheless, Eucalyptus has shown itself, on the one hand, as an effective means of pooling resources in such a highly heterogeneous environment, its tools make it easy to scale the environment, and on the other hand, to reconfigure existing resources, which is very important for university networks with their heterogeneous load. Compared to Eucalyptus, OpenNebula requires more platform support and dynamic scalability management of virtual machines on clusters.

For hybrid clouds, it provides on-demand access and more flexible management mechanisms like Amazon EC2. Continuous Delivery: automates the entire software release process. It can consist of several stages. Deployment to production is done manually.

Jenkins has several useful third party projects. The first is the Configuration as Code plugin. It simplifies Jenkins configuration through readable APIs that are understandable even for administrators without deep knowledge of the tool. The second is the Jenkins X system for the cloud. It accelerates the delivery of applications deployed on a large-scale IT infrastructure by automating some of the routine tasks.

Big data cloud services can reduce the effort of setting up and configuring your own clusters, and make sure you pay for what you use. The biggest problem will always be with the location of the data, since sending data is slow and expensive. The most effective big data solutions in the cloud will be those in which data is also collected in the cloud. This is a reason to look at - the incentive of EC2, Azure, or AppEngine as the main application platforms, and an indicator that PaaS competitors like Cloud Foundry and Heroku will need to prioritize big data.

PART 3
IMPLEMENTATION OF CONTINUOUS SOFTWARE DEVELOPMENT USING CLOUD TECHNOLOGIES

3.1. Continuous deployment and integragion system topology analization

Over the past 30 years, the IT market has tried a lot of software development techniques. It all started with a waterfall model - a complex, multi-step process that, as the time went on, the less suited the rapidly changing market. In the waterfall model, each developer is busy with their own piece of code. Integration occurs at the final stage of the build, while identifying errors entails an unpredictable release delay. A complete assembly cycle can take more than one month. In the same web development, where changes to the code need to be made often and quickly, the waterfall model is of little use.

The waterfall model has been replaced by jenkins development methodologies such as Jenkinks. Due to its convenience and versatility, it is used in retail, finance, IT and not only, both small and large businesses. Jenkins, however, has a number of drawbacks, which stand out especially against the background of another equally popular CI / CD methodology - Continuous integration & Continuous delivery. In this post, we will make a short comparative analysis of these two techniques, and also tell you how to properly implement CI / CD without unnecessary costs and nerves - based on own experience.

Jenkins methodology is based on an iterative approach: the team works in short cycles - sprints, releasing the product in stages. Jenkins is good for those areas of development that require quick and regular, up to hundreds of times a day, releases of working and updated software versions: mobile apps, GameDev, web development, e-commerce, etc. Build and release rates dictate the length of the sprint, which usually takes from one to two weeks - and this is precisely one of the problem areas of Jenkins, in which practice diverges from theory.

Increasing the speed, teams often neglect the necessary stages: some updates or fixes are not tested - for example, regression testing is not carried out, the release is made "at the last moment". Hence - the growth of customer complaints about the service performance. Skipping the required build steps discredits the very idea of Agile.

Jenkins process is built by a team lead, or senior developer, who is competent in the development area, but who may have gaps in infrastructure administration. (After all, the developer is required, first of all, to have a thorough knowledge of programming languages and applied tools for building, testing and deploying code.) With an increase in the number of tasks, the team leader cannot always adequately assess his capabilities and the development process in the complex. Short, for example weekly, sprints are another factor that complicates project management: when you constantly plan, consult and inspect, that is, you are busy with managerial work, there are 2-3 days left for the main tasks.

The left side of Fig. 3.1. shows a traditional deployment process. The developer commits the set of changes to a SCS and building server starts to build a job.

The right side of Fig. 3.1 shows the continuous deployment process. After master is added, the Jenkins master appears

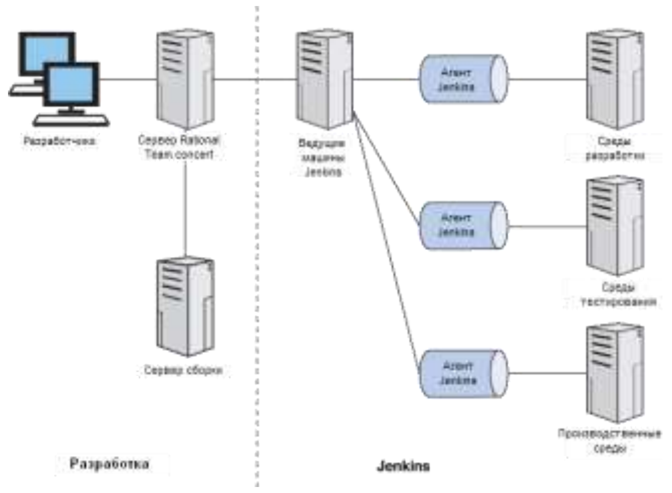


Fig. 3.1. Continuous deployment system topology

Jenkins focuses on the interaction between the business and the development team, while the CI / CD methodology focuses on the interaction between developers and the administrator of the infrastructure on which the code and product live. The proposed CI / CD principles gave impetus to the development of automation tools and simplification of the code building process. Major software vendors - including GitHub and GitLab - in addition to working with the repository, have implemented Kanban boards that visualize the workflow, and have automated deployments (software deployment) [19].

CI / CD Continuity is also about continually improving the build process. In the long term, the budget is saved, the quality of the product improves, the need to work in a constant rush disappears, and the release of the product is accelerated. Understanding the best practices accumulated by development administrators, the same DevOps engineers, is a valuable product in itself.

The next important step is the use of automated testing in order to cover the functionality of the developed code (both software and infrastructure) as much as possible. In other words, having a deployed infrastructure, but without automated testing, the bottleneck of the process will be the timely verification of the functionality. Automation of the testing process should begin with the actual writing of the software code (unit testing), the application of primary tests on the server that is responsible for building the software, and a test of the server configuration. This will reduce the load on the software quality team and significantly reduce the time it takes for software to pass through the pipeline.

The purpose of the scientific research is to create a centralized, highly accessible database for managing configurations of the development team, as a source of current reliable information about all components in the IS, for the subsequent automation of technological processes for managing changes, incidents, continuous monitoring and recovery operability of computing elements in the information cloud.

3.2. Jenkins installation and additing nodes for master slave

In this article, we'll walk you through how to install Jenkins and, using a Java web application as an example, show you how to automate the build process and deliver it to the Jetty web server. CD provides automatic delivery and deployment of build results to target servers. The source code for the web application is in the git repository, and Maven is used as the build tool:

- the agent directive uses the docker property where you can specify;

- the name of the assembly container according to your naming policy;

- arguments needed to run the build container, where in our case we mount the current directory as a directory inside the container.

Jenkins is one of the popular tools that allows you to implement the approach of continuous integration (Continuous Integration, CI) and continuous delivery (Continuous Delivery, CD) of software.

The following will act as slave agents:

- Standalone (separate) Linux server

- Docker container launched at the request of the Jenkins master

- Standalone (separate) Linux server

- Docker container launched at the request of the Jenkins master

To determine the specific slave / agent on which the assembly should be built, labels will be used, both on the slave / agents side (each such agent will have a label), and in the options of the assembly itself, which needs to be assembled [20].

In particular, in this example, we will collect on the Slave server only those assemblies whose labels coincide with the label of the Slave server itself (linux-slave1), which we can see on the Fig.3.2.

CI greatly facilitates the integration of scattered copies of a project with the main development branch, allows you to set up automatic project building and run test CD provides automatic delivery and deployment of build results to target servers.

In this article, we'll walk you through how to install Jenkins and, using a Java web application as an example, show you how to automate the build process and deliver it to the Jetty web server.

The source code for the web application is in the git repository, and Maven is used as the build tool. To add a slave node to Jenkins, you can add each machine by hand, but because of this, you need to constantly go to the machine and perform the same operations, and this leads to the idea that everything can be automated as much as possible.

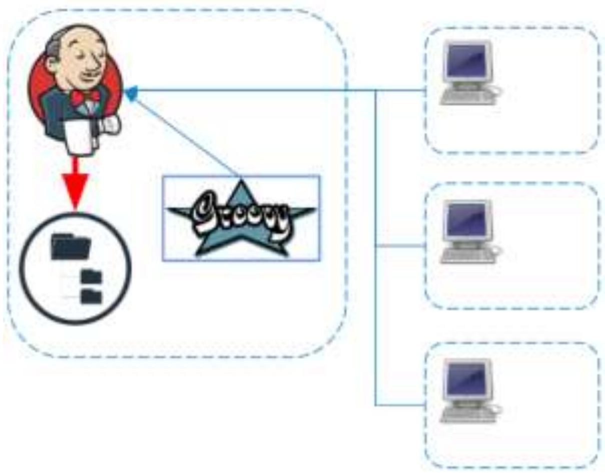


Fig. 3.2. Representation of load distribution in Jenkins

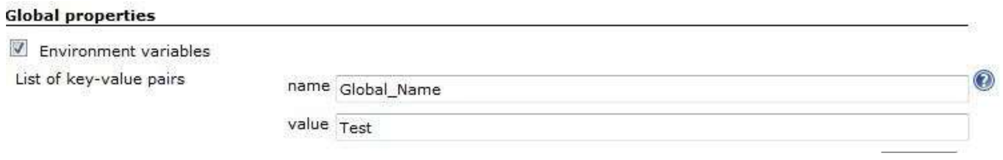
Jenkins currently supports over 1000 plugins. These plugins can be divided into categories (plugins for source code management, for generating reports, for creating tools, etc.). Using plugins, you can monitor, deploy or customize various jobs in Jenkins. To manage plugins, open https: // Host: 8080 and select Manage Jenkins. There are four tabs: updates: installed plugins for which there are updates, available: plugins available for installation. Installed: installed plugins. Advanced: plugin management. Installing plugins over the Internet. If the Jenkins master does not have Internet access, the plugins can be installed manually. Master machine has Internet access, the plugins are easy to install. Select plugins to install on the Available tab. You can uninstall plugins on the Installed tab.

Jenkins is one of the popular tools that allows you to implement the approach of continuous integration (Continuous Integration, CI) and continuous delivery (Continuous Delivery, CD) of software.

CI greatly facilitates the integration of scattered copies of a project with the main development branch, allows you to set up automatic project building and run tests.

In the global properties, you configure the environment variables (defining the property name and value) or the location of the tools: go to http://<Host>:8080 and select Configuration. These properties can be used in all Jenkins projects.

You can reference environment variables in projects. Select the environment variables field and define the name and value of the variables as shown in Fig. 3.3.


Global properties

☑ Environment variables

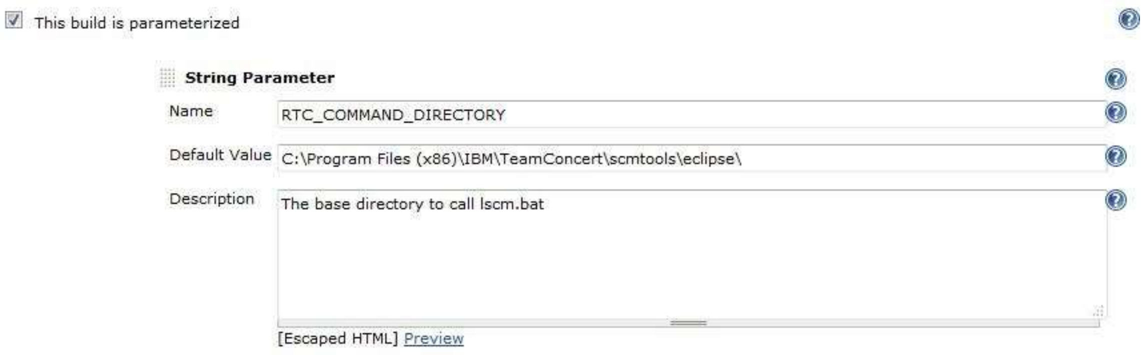
List of key-value pairs

name: Global_Name

value: Test

Fig. 3.3. Defining environment variables

Local properties are only available within the project. When configuring the project, select the This build is parameterized option, as shown in Fig. 3.4. It allows you to add parameters as name-value pairs. These parameters can be used as local project properties.


☑ This build is parameterized

String Parameter

Name: RTC_COMMAND_DIRECTORY

Default Value: C:\Program Files (x86)\IBM\TeamConcert\scmtools\eclipse\

Description: The base directory to call lscm.bat

[Escaped HTML] Preview

Fig. 3.4. Setting local properties

Practical Application: Environment Design and Continuous Deployment Process. The goal of continuous deployment is to provide an efficient and high-quality process for developing, testing, deploying, and launching software into production. In continuous deployment, every change to any part of the software system (infrastructure, application, or data setup) is continually pushed into production through a dedicated deployment pipeline. Process implementing continuous deployment requires a fast, automated process for deploying changesets. The deployment process consists of several steps. The standard process is as follows: the developer makes changes to the code, source control tools build, automatic testing is performed, and the assembly is installed.

3.3. Differences in project builds that affect on process of continuous software development

And already in the build steps, we indicate which commands to execute inside the Docker build agent. It can do anything, so I also launch the application deployment using ansible.

Thanks to this method, we solved the following problems:

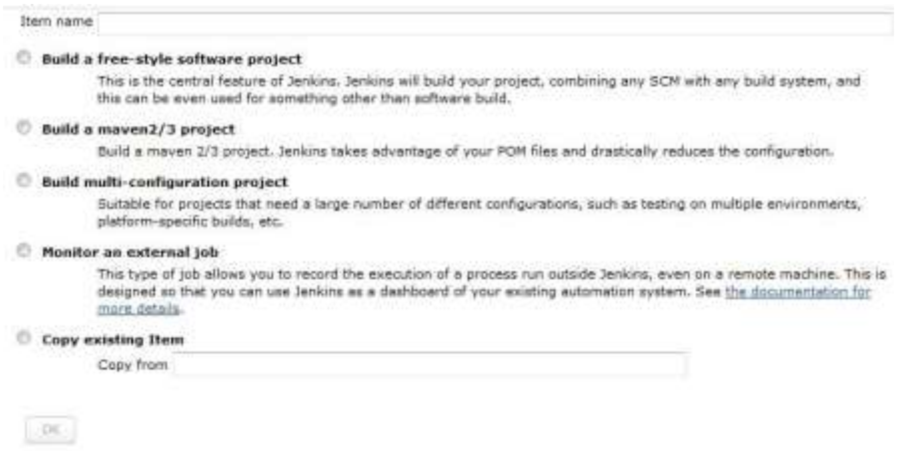
- environment build configuration time is reduced to 10 - 15 minutes per project;
- completely repeatable environment for building the application, since you can also build it on the local computer;
- no problems with conflicts between different versions of assembly tools;
- always a clean workspace that does not clog.

The solution itself is simple and obvious and allows you to get some advantages. Yes, the entry threshold has risen a little compared to simple commands for assemblies, but now there is a guarantee that it will always be assembled and the developer himself can choose everything that is necessary for his assembly process.

You can also use the Jenkins + Docker image I have collected. All sources are open source and can be found at rmuhamedgaliev / jenkins_docker.

During the writing of the article, a discussion arose about using agents on remote servers, so as not to load the master node using the docker-plugin plugin. But I'll talk about this in the future.

Using the Master (the main and only server on which Jenkins is directly installed) together with agents (slave servers) allows you to build on these agents, thereby reducing the load on the master server, to perform builds on various software / operating systems , execute different steps of the same build in parallel on different Jenkins agents (for example, running parallel tests for different WEB browsers (Chrome, Firefox, Opera, etc.) or running a large number of integration tests can be divided, for example, into 4 agents, thereby reducing the total execution time of all tests theoretically by 4 times). Consider setting up two types of Jenkins slave / agent, which, for example, will build a Java project using maven.


Item name

Build a free-style software project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Build a maven2/3 project
Build a maven 2/3 project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Build multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

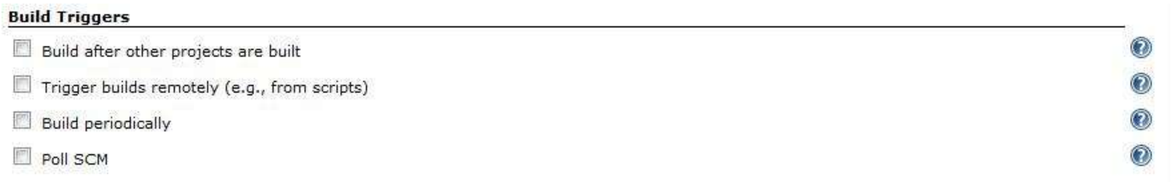
Monitor an external job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system. See the documentation for more details.

Copy existing Item
Copy from

OK

Fig. 3.5. Variable types of automation our jobs for software development

For example, a build of a project can be called by email or a build request can be sent from a script. Periodic Build: Create a schedule to periodically build projects in accordance with the configuration. SCM poll: This option builds the project by making changes to the source code. In this case, specify how often Jenkins should poll the source control system. If there are changes to the source code, the project is built.


Build Triggers

☐ Build after other projects are built
☐ Trigger builds remotely (e.g., from scripts)
☐ Build periodically
☐ Poll SCM

Fig. 3.6. Build launch parameters

Jenkins Master connects via SSH and copies the jar file to Slave to the directory specified by the parameter

"The root of the remote file system" (in our case, this is the home directory of the jenkins user - the / home / jenkins directory) At the same time, build settings, its execution logs, test results, artifacts, etc. everything is stored on the master server, the agent only stores the build runtime (workspace), in which the build is assembled directly.Inside the container, execute the commands necessary to create a group and user of the jenkins user, set a password for it (this login (jenkins) and password (set at this stage) will be used when creating Credentials in Jenkins. Install the software that is used when performing the build, for example java, git, maven, and also ssh server [21].

According to the label in the node field, the build will be launched on Jenkins-slave with the label docker-maven-build-slave, this label that corresponds to our docker container configured in the previous step (in point 5)

3.4. Analyzing continious delivery using ansible

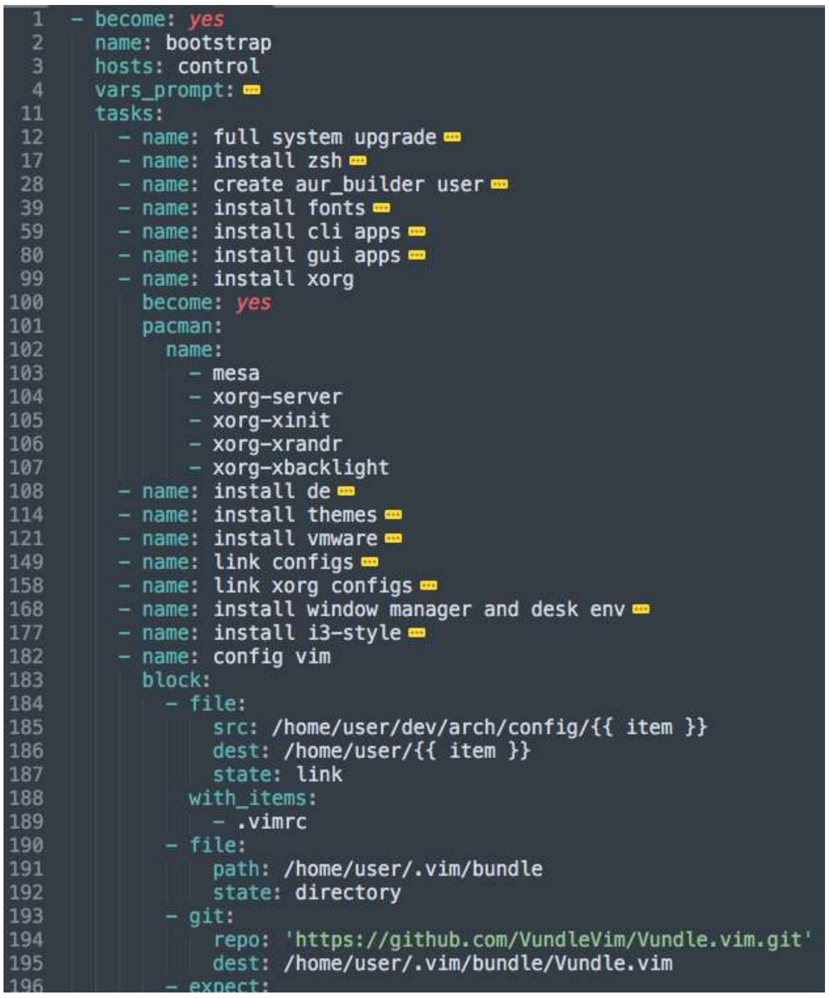
Test Driven Development (TDD) and other related techniques further enhance the efficiency of new ways of working. And if the slogan "release faster and more often" has become synonymous with progress, then CI / CD can be your path to true bliss. Agile symbolizes movement forward, and following this path offers significant benefits to organizations, the key to which is automation. Below, we'll show you why Red Hat's Ansible and Ansible Tower are ideal for orchestrating IT systems within today's software development processes [22].

Tasks can be as many as you like, and Ansible will walk through each visitor's eyes. Look at the butt of a simple config - there are descriptions in one file (some of the blocks are twisted for a glance). Of course, more configs are manually divided into files and tags, but for our staff there are enough simple ones.

Ansible is an open-source configuration management tool that can help you automate tasks and quickly deploy applications. It is an easy-to-install, highly efficient and powerful tool.

It is a free tool that can run on multiple operating systems like MAC, Linux, BSD, etc. Besides the free version, there is also one enterprise version called Ansible Tower, which is commonly used to get the most out of various industries. ... Blog highlights include as shown below.

Each task has its own parameters (for example, a list of packages to install, or paths to files that need to be copied). More details on how to write different types of problems are below (Fig.3.7.).



```
- become: yes
  name: bootstrap
  hosts: control
  vars_prompt:
  tasks:
    - name: full system upgrade
    - name: install zsh
    - name: create aur_builder user
    - name: install fonts
    - name: install cli apps
    - name: install gui apps
    - name: install xorg
      become: yes
      pacman:
        name:
          - mesa
          - xorg-server
          - xorg-xinit
          - xorg-xrandr
          - xorg-xbacklight
    - name: install de
    - name: install themes
    - name: install vmware
    - name: link configs
    - name: link xorg configs
    - name: install window manager and desk env
    - name: install i3-style
    - name: config vim
      block:
        - file:
            src: /home/user/dev/arch/config/{{ item }}
            dest: /home/user/{{ item }}
            state: link
          with_items:
            - .vimrc
        - file:
            path: /home/user/.vim/bundle
            state: directory
        - git:
            repo: 'https://github.com/VundleVim/Vundle.vim.git'
            dest: /home/user/.vim/bundle/Vundle.vim
        - expect:
```

Fig. 3.7. Configuration for deploying a processing system environment

Achieving zero downtime is possible, but it requires the right tools - those that provide advanced, multi-tier and multi-stage orchestration, such as Ansible.

Serial updates flag for temporarily disconnecting a host to disable availability monitoring for it during the update period. Software level - return to the pool - turn on monitoring" easily without the participation of an operator.

It can handle a variety of Inventory files, making it easy to test rolling upgrade scripts in staging environments before running them on production servers. To do this, you just need to simulate a production environment in a test environment.

Test During Deployment. The more opportunities, the higher the responsibility. Automating continuous delivery processes dramatically increases the risk of deploying a failed configuration to all nodes in the system. To mitigate the risks, Ansible suggests inserting benchmarks into scripts that will abort the rolling update if something goes wrong. You can deploy arbitrary tests using Command or Script modules to check various conditions, including the status of services, and even create such tests as separate Ansible modules.

Creating our playbook. We already know that for the most part a playbook is a collection of tasks of varying degrees of nesting. Now let's learn how to write these tasks. The main thing to learn is that each task requires the use of a module. A module is an actor that can perform actions of a certain type. For instance: install and update packages; create and delete files; clone repositories; ping hosts; send messages to Slack.

From now on, let's agree that we do not separate the master and target machines, we perform all actions on one machine. For remote hosts, the procedure is much the same. Before running the script, tell Ansible a list of hosts to perform operations on. Usually, the list by group is specified in the / etc / ansible / hosts file.

For each task, we pass the module and a set of parameters for its launch, for example, a list of packages to install. There are a lot of modules, you can see their full list here. In the meantime, let's go through the ones that we need. Ansible is ideally suited for CI / CD tasks, including multi-level multi-stage orchestration of one-to-one software updates with zero downtime, thanks to its extensive set of features and tools. This feature of Ansible, plus its unique architecture and the absence of software agents on target hosts (increasing security and eliminating the need to manage the control system itself) make it easy to describe and easily automate complex deployment processes. Previously, such processes were performed exclusively by operators (both manually and with partial automation) and required coordinated actions of all participants in the process. Ansible implements full autopilot mode here. Autopilot deployment paves the way for agile development, allowing software changes to be made faster, with fewer bugs, and significantly less context switching. No planned downtime and, more importantly, no human error - that's exactly what mission-critical enterprise applications and queuing web systems need.

3.5. Job management in Jenkins

Before proceeding with the main part, it is necessary to define the terms used. Currently, Continuous Integration is one of the practices used in Agile methodology. In such methodologies, it is successfully combined with other practices, such as unit testing, refactoring, coding standard. But even without them, you can still benefit from continuous integration. From the definition of Continuous Integration, its most important idea follows - to find an error at an early stage is easier for development and cheaper for business. Therefore, if sufficiently large changes are made to the project, it is imperative to conduct testing (unit tests, automated tests).

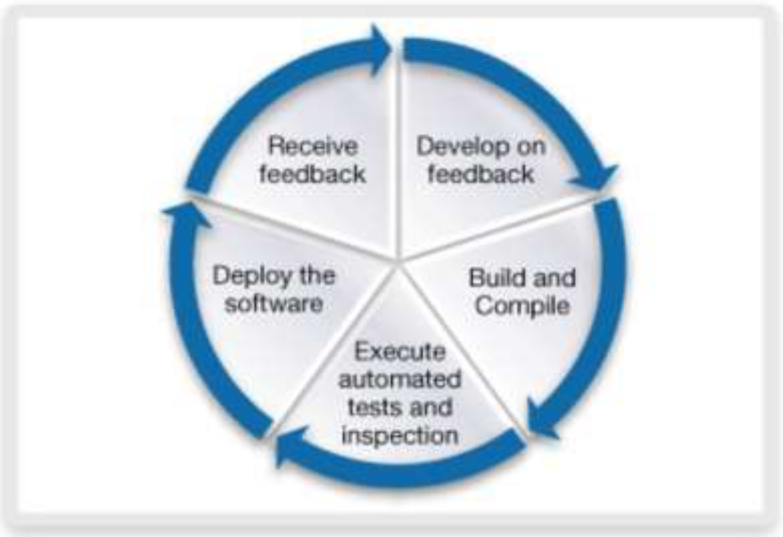


Fig. 3.8. Analazing a process of Continious delivety

It's worth noting that there can be multiple run modes: the first tests are run after each commit - this is perfect for unit tests and inconsistent for end-to-end, so I'll explain below.

It is necessary to block the restart of the very first pipeline, since it is executed on the same VM. The second pipeline should work without restrictions, regardless of whether the first one is running or not.

Didn't work with Pipeline, only Job DSL - in waitUntil, can I use a Groovy script with access to Jenkins classes. Thus, you can probably write logic in the first job (compile-1), which will look at general test jobs (testA, testB, testC) and check if they are executed with the same IP parameter. And based on this, postpone the start of the next one (deploy-1). But in this case, the first job will be executed at an arbitrary time, right? Not very good both from the visual side ("oh, here some job is stuck") and from the side of getting metrics (build duration on Fig.3.9.).

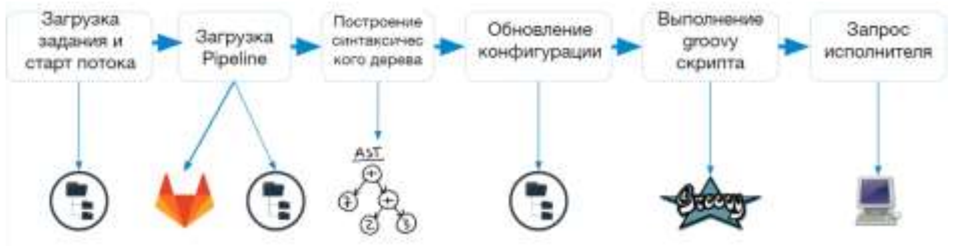

Загрузка задания и старт потока
Загрузка Pipeline
Построение синтаксического дерева
Обновление конфигурации
Выполнение groovy скрипта
Запрос исполнителя
AST

Fig. 3.9. Implementation of project bild

When the pipeline jobs start, Jenkins automatically start to execute the jobs, which are written on our node if thete aren't quere. The most popular and developing now is the selfhosted project.

Can use Docker to work fine. The whole circuit looks like this(Fig.3.10):

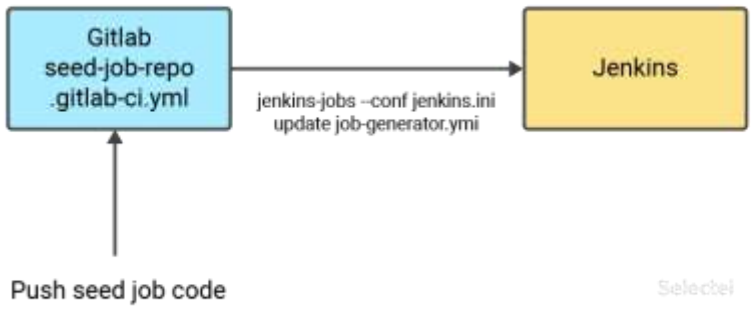


Fig. 3.10. Push seed job

In the pipeline, we will build a docker image, which will then be uploaded to the local docker registry. The image file will be taken from another project.

For some reason, if I use a regular expression, the $ GIT_BRANCH environment variable seems to contain an irrelevant match (for some reason, a branch of the previous job assembly). And from the looks of it, the change trigger in the SCM works indefinitely.

When using a webhook, the $ gitlabB environment variable is available to me, and the regular schedule itself is already registered in the webhook job settings.

also, just in case, in the shell script that I run within a specific job, I check for the presence of $ gitlabB this was necessary to prevent the job from continuing the build if $ gitlabBranch is missing (for example, if you run it manually). Earlier (when I thought that everything worked like that), manual build simply started the build of the last branch that matched on the regular basis. If I figure out what's the matter and where I made a mistake, I'll share it.

In a project named sample-project, create a Jenkinsfile with the following content:

```
node {
 def app
 stage1 ('Clone the repository from Git') {
       checkout([$git: 'SCM', branches: [[name: '*/node']],
       doGeneratemodulesConfigurations: true, extensions: [],
       submoduleCfg: [], userRemoteConfig:
       [[credential-Id: 'gitlab-repo-cred',
       url: 'git@gitlab-my-prod-env:test-cont /dockerimage-file.git']]])
 }
 Stage2(Building image in Docker container') {
       apps = dockers.building("dockerimage")
 }
 Stage3 'Testing for problems') {
       if docker inspecting doc-image > /dev1/nhom; then
      echo 'Image№  does not  existed!'
      exit 1
 }
 Stage4(Pushing in prod) {
       echo 'Pushing in prod!'
       docker.withRegistry('https:// my-prod-env:5310', 'registry-credentials') {
      app-push("${env.ENV _NUMBER}")
      app-push("The latests")
       }
 }
```

```
Stage5 (Clean existing img) {
  sh -l " rmi dockerimage"
}
```

As a result, the finished pipeline will look like this in the blueocean interface in Jenkins look like imagine in Fig. 3.11.

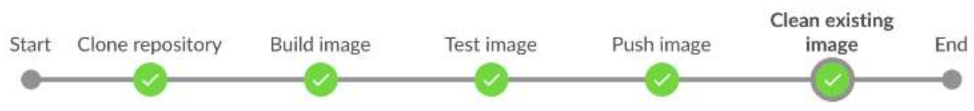
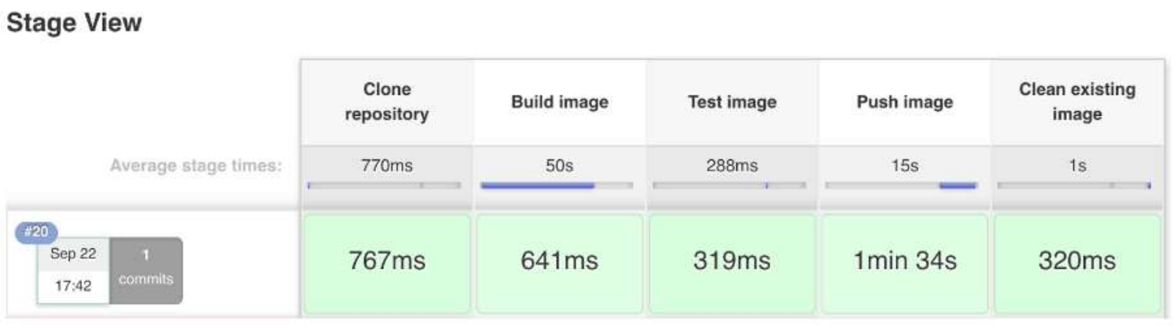


Fig. 3.11. Pipeline stages results representation in blueocean interface

Or like this in the classic interface:



Stage View

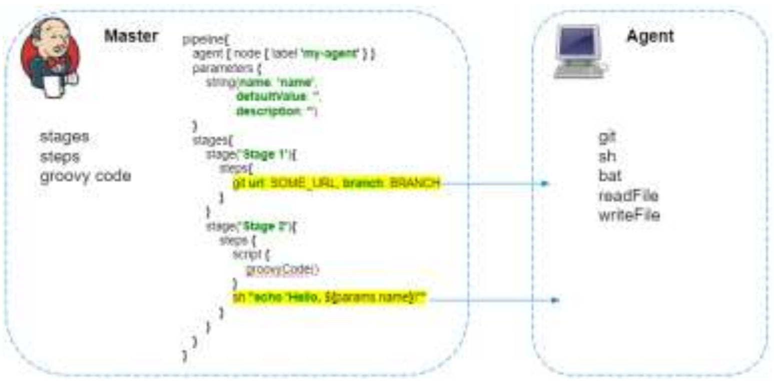
	Clone repository	Build image	Test image	Push image	Clean existing image
Average stage times:	770ms	50s	288ms	15s	1s
#20 Sep 22 17:42 1 commits	767ms	641ms	319ms	1min 34s	320ms

Fig. 3.12. Stage view of our system of continuous software development

To run this task automatically, we can add a webhook in the project with docker-image, where after submitting the changes to the repository, this task will run.

3.6. Testing Our jobs for complitness and working with logs and artifacts to spped up our infrastructure

There is already little room for QA who do not understand the technological aspects of the application, CI / CD processes. A testing strategy is a plan that will allow you to work with a minimum investment of time and therefore money. A testing strategy for DevOps should not sound like "We use Protractor and Jenkins." This is not a plan. This is just a listing of the tools you are using. The main questions that the plan should answer are why and why. We must answer ourselves the following questions: The first thing we should start with is to implement the TDD concept.



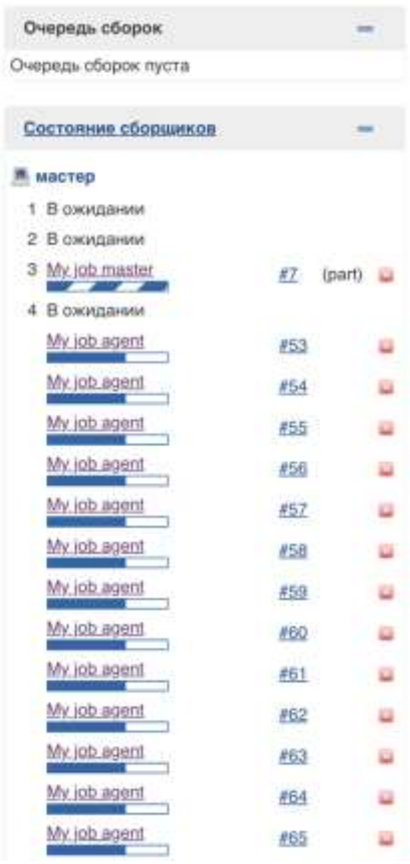
```
pipeline{
  agent { node { label 'my-agent' } }
  parameters {
    string(name: 'name',
           defaultValue: '',
           description: '')
  }
  stages{
    stage('Stage 1'){
      steps{
        git url: SOME_URL, branch: BRANCH
      }
    }
    stage('Stage 2'){
      steps {
        script {
          groovyCode()
        }
        sh "echo 'Hello, ${params.name}!'"
      }
    }
  }
}
```

Fig. 3.13. Representation of connection master node with slaves

I must say right away that TDD is not absolutely the right choice, but it does contribute to writing tests as such. That is, developers must write tests before coding begins. For developers, any save action should trigger the launch of unit tests. In addition, the engineer should not be able to flood his code with broken tests or when the test coverage falls below some limit. When pushing code into a feature branch, integration and unit test suites should be run.

As you can see, each time we run more and more resource-intensive tests, while cutting off errors that may be associated with different factors. You have a test pipeline in which your product moves from one test to another. conclusions Automated testing is one of the essential practices of the DevOps culture. It allows you to control the original product and quickly eliminate errors, from the development process to the process of deploying to the production environment. I also want to say a big thank you to everyone who leaves comments on my articles, to all the people who are actively involved in the discussion of this series of articles.

They determine the status of the build and the ability to merge it into the main dev branch. With a broken build, there should be no possibility of merging, and the committer should be notified that his commit has broken the branch. Next comes the merge in master or stage brunch and build there. This requires running all test suites, including E2E. You can notice that they are executed on the master, but they do not affect the regular executor pool:



Очередь сборок

Очередь сборок пуста

Состояние сборщиков

мастер

1 В ожидании

2 В ожидании

3 My job master #7 (part)

4 В ожидании

My job agent #53

My job agent #54

My job agent #55

My job agent #56

My job agent #57

My job agent #58

My job agent #59

My job agent #60

My job agent #61

My job agent #62

My job agent #63

My job agent #64

My job agent #65

Fig.3.14. Job build queue in system of continuous software development

This test set determines if your product can be shipped. In the event of a failure, all members of the dev-team should be notified that the delivery is pending and the reason that caused this error. If all the suites on the test environments went well, the product is delivered to production and the E2E test suites are run again on the production environment. In addition to the above test suites, performance, penetration and other non-functional tests should be run in production. I needed to configure monitoring of the Redis server. I dug a little in the code of the available plugins.

If something went wrong, a rollback is carried out and everyone involved in the deployment and development process is notified. Note that the rollback must also be tested by E2E and other tests.

The result of our pipeline presented in table 3.1.

Table 3.1.

Jenkins build test result

Stage of continuous development	Time in ms
CLONE REPOSITORY	767
BUILD IMAGE	641
TEST IMAGE	319
PUSH IMAGE	1.34
CLEAN EXISTING IMAGE	320

Zabbix is a great product for administrators of large hardware and software systems. It is so good that it can be used not only by large businesses, but also by medium and small businesses, and even in a pet project. In general, I have little experience with Zabbix and I can safely recommend it for use. However, I cannot say that I understand the "philosophy of Zabbix". Despite the extensive detailed documentation in Russian, it was difficult for me to dive into the world of Zabbix - I got the feeling that the developers and I call the same things by different names. Perhaps because Zabbix was created by admins for admins, but I am more of a developer and user.

Nevertheless, to run Zabbix and to monitor the main parameters of computer systems (processor, memory, etc.), the skills of an ordinary linux user are enough. There are a large number of plugins from third-party developers that extend the capabilities of Zabbix. For my needs, found out that the architecture of Zabbix makes it easy to connect to monitoring any parameters of information systems that can be expressed in numerical form.

Under the cut - an example of a Zabbix plugin with my explanation on Zabbix terminology. To some, this example will seem naive, but to someone it will help you to more easily get used to the concepts. In any case, Zabbix is large enough to feel from different angles. In the event of an event (triggering), the server can perform an action. For example, send an email notification to the specified address ("Problem: host is unreachable for 5 minutes"). Also, the action can be performed if the trigger returns to its original state ("Resolved: host is unreachable for 5 minutes"). All events (trigger switches) are logged on the server side.

I find Zabbix a great tool, especially after I discovered the ease of adding my own monitoring parameters (items). By and large, it is enough to add one file to the server with the agent (userparameter_XXX.conf) with a shell command to collect data and configure the Zabbix server to receive this data through the web interface. And that's all - you can accumulate data, build graphs, analyze changes and create triggers that react to these changes.

Zabbix is considered to be one of the most advanced tools for remote monitoring of hardware and software resources. The system successfully allows you to solve problems of monitoring network activity and server health, as well as to warn of potentially dangerous situations. Thanks to the built-in analysis and forecasting mechanisms, Zabbix can become the basis for creating a complete strategy for the effective use of IT infrastructure in companies of any size.

Conclusions on the Third Part

For continuous deployment, it is important to set up your environment correctly. It determines the effectiveness of the DevOps system and what can be done during continuous deployment.

This article provides information about the Jenkins platform and demonstrates:

– How to set up a continuous deployment environment with Jenkins;

– apply this knowledge within a continuous deployment environment;

– implement a continuous deployment environment with Jenkins.

It is an automation environment that performs repetitive tasks. Jenkins can execute and control the execution of commands on remote systems, as well as anything that can be done from the command line. Jenkins integrates email, TestNG and other tools with helper plugins.

Once installed (see sidebar), Jenkins is accessed through a browser at http: // yourjenkinsmasterhost: 8080. Jenkins supports master-slave mode. The work of assembling projects is delegated to several slave nodes.

DevOps and testing strategy The DevOps culture encourages frequent releases. Frequent releases are insurance against breaking your application after irreparable improvements by the developers, since you will roll out fewer changes at a time. Frequent releases, however, require more QA work. That is, if you pour out almost every hour, then your quality engineers will simply stop going home, eating, drinking and leading a social life. DevOps culture suggests minimizing manual QA work in order to be able to release as often as possible, which is safer and more stable. In DevOps culture, the role of QA engineers shifts from testers to people who monitor the quality of the project and help developers write automated tests, develop a testing strategy.

CONCLUSIONS

The primary goal of the Graduation Project – “System of continuous software development using cloud technologies” – is the design system with a help of CI/CD pipeline to provide Continious software development and fast integration. The topicality of the graduation project is the necessity to analyze the existing platforms, methods and technologies of continious delivery, choose the most suitable for the given task and apply it with recommendation of further improvement. During the graduation project preparation, a great volume of information was collected and analyzed.

Part 1 of the graduation project contains the detailed description of the actuality of cloud services and main benefits of their use. Already today, continuous software development is very important for business representatives, because in this case the number of releases increases. All this is due to the fact that the process is automatic and without the help of an additional system manager to perform work faster and without human influence.

Despite a number of shortcomings, cloud technologies remain popular and active used by both teachers and students. Modern information technology combined with teachers and students are enabled by uninterrupted connection to the Internet use an effective tool to organize the learning process as a solution learning tasks are greatly simplified. Possibilities of providing multilevel access to training resources located in the cloud environment allow accurate dosing access and provide documents only for the intended use. Thus, cloud technologies have significant potential and open wide prospects not only for educational institutions but also for the individual who is interested in receiving quality education, because they create an opportunity for lifelong learning support for mobile technologies and social networking services, make the process itself learning interesting and interactive.

Part 2 contains analysis of hyperscalers, cloud technologies and continuous system of continuous software development highlighted a number of functions that have become available with our system:

– save time by reusing code and quickly deploying projects;

– obtaining the expected result from the deployment;

– independence of the project from the environment;

– immutable infrastructure and easy migration;

– possibilities of the Infrastructure as Code approach;

– the ability to analize a logs of mistakes and and quickly fix the error;

– distribution of all data to different repositories automatically

– ability to interact all project team as one whole.

Diploma work presents a solution for building a continuous software processing using various cloud services and technologies. During the review of the work and construction of a system of continuous processing, a large number of problems were solved:

– analyzed representative types of computers on different platforms;

– several levels of protection have been established for the Jenkins job builder.

– worked with VPC in AWS and configured the SSH connections in different types on EC2 instances;

– virtualized our project builds in containers;

– configured continious software development with a help of scripts or commands;

– Infrastructure as a Code analyzed;

– automated software delivery, configuration management, and application deployment.

Part 3 describes implementation of continuous coftware development. From here it is obvious that it is not enough to simply study technologies Gitlab CI, Doker, Ansible, terraform, etc. on which CI / CD is built. Therefore, during the course we paid special attention to the development of practice. Project work is the final part of the training, where you have to implement CI / CD processes for any opensource project of your choice.

In addition to these advantages, there are disadvantages and problems that hinder the implementation of cloud computing, namely:

– the inability to work with cloud services without a constant connection to the Internet;

– complex or impossible process of transition from one cloud service provider to another;

– lack of a single international legal regulation in the field of cloud computing and cloud information processing;

– issues of protection of user information processed and stored in the cloud.

With all the benefits of using cloud computing, there are many security issues that are not well analyzed today and are still under discussion. As shown in the article, the main problem that has not been solved in the field of cloud computing today is the trust of users in the service provider. This problem is acute not only for companies and enterprises that use third-party suppliers, but also for ordinary users, whose personal data also need protection and security guarantees. If in the case of a large enterprise it can protect itself from threats by conducting a security audit of the cloud service provider and analyzing the risks and threats of information security, as well as insuring them or creating its own private cloud, then small companies or ordinary users do not have this opportunity.

Therefore, it is necessary to implement mechanisms to control cloud service providers at the international or national level, in order to conduct a security audit and verify their compliance with international or national standards and the conditions imposed on them.

REFERENCES

1. Recommendation ITU-T, 2014, Cloud computing – Functional requirements of Infrastructure as a Service, Y.3513, c.24.

2. Amazon Elastic Compute Cloud documentation [Електроний ресурс]. - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html (звертання 2 грудня 2020).

3. Satoshi T., Yuichi T., Big Data Processing in Cloud Environment, Fujitsu Inc., Tokio, pp. 157.

4. Recommendation ITU-T, 2016, Functional Architecture of NaaS, Y.3515, c.11.

5. Гудкова И.А., Масловская Н.Д., Вероятностная модель для анализа задержки доступа к инфраструктуре облачных вычислений с системой мониторинга, 2014.

6. Эберхард Вольф, Continuous delivery. Практика непрерывных апдейтов. Учебник. -М.; 2018. – 652-654 с.

7. Kanikathottu H., AWS Security Cookbook: Practical Solutions for Managing Security Policies, Monitoring, Auditing, and Compliance with AWS, Packt Publishing Ltd., Birmingham, pp.326.

8. Chirinos R. Bringing technologists together to connect, collaborate, and learn about AWS, Library Published in: Science Computing Conference (SCC), 2019 European Date of Conference: 6 May 2019. 110-123 pp.

9. Тестируем облачные платформы из Топ-3 2017. [Электронный ресурс] // Хабр: [сайт]. [2018] - Режим доступа: https://habr.com/ru/post/328916/.

10. Облачные технологии: основные понятия, задачи и тенденции развития [Електронний ресурс]. - http://swsys-web.ru/cloud-computing-basic-concepts-problems.html (звертання 25 листопада 2020).

11. Дроздова И. И. Безопасность облачных хранилищ // Молодой учёный. - 2017. - С. 16-18. [Электронный ресурс] // Молодой учёный: [сайт].

[2017] - Режим доступа: https://moluch.ru/conf/tech/archive/286/13236/ (дата обращения: 11.12.2020).

12. Широкова, Е. А. Облачные технологии // Современные тенденции технических наук : материалы I Междунар. науч. конф. Уфа, октябрь 2017. С. 30-33. - [Электронный ресурс] // moluch.ru [сайт]. [2017] - Режим доступа: https://moluch.ru/conf/tech/archive/5/1123/ (дата обращения: 14 жовтня 2020).

13. Hyperscalers Emerging From ‘Hype Phase’ [Электронный ресурс] // EnterpriseAI: [сайт]. [2017] - Режим доступа: https://www.enterpriseai.news/2017/04/12/hyperscalers-emerging-hype-phase/ (звертання 21 листопада 2020).

14. Installing Jenkins [Электронный ресурс] // Jenkins cd [сайт]. [2020] - Режим доступа: https://www.jenkins.io/doc/book/installing/.

15. Евстратов, В. В. Контейнеризация как современный способ виртуализации // Молодой ученый. — 2020. — № 49. — С. 7-9. — [Электронный ресурс] // Молодой учёный: [сайт]. [2020] - Режим доступа: https://moluch.ru/archive/339/76072/ (дата обращения: 13 листопада 2020).

16. Олифер В.Г., Олифер Н.А.. Компьютерные сети. Принципы, технологии, протоколы – Учебник. – М.:, 2016. – 672с.

17. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: НАУ, 2017. – 63 с.

18. The Industry Standard For Cloud Applications [Электронный ресурс] // Cloud Foundry [сайт]. [2020] - Режим доступа:https://www.cloudfoundry.org/ (звертання 17 листопада 2020).

19. Облачные вычисления (Cloud computing), 2015 [Електронний ресурс]. – http://www.tadviser.ru/index.php/Cloud_computing (звертання 21 листопада 2020).

20. Virtualization [Электронный ресурс] // vmware [сайт]. [2020] - Режим доступа: https://www.vmware.com/solutions/virtualization.html.

21. Ansible project [Электронный ресурс] // Ansible Documentation [сайт]. [2020] - Режим доступа: https://docs.ansible.com/.

22. Литвинова С. Г. Поняття та основні характеристики хмарного середовища // Інформаційні технології і засоби навчання, 2014. – Том 24. – №4. – С.14-34.

23. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення. – Чинний від 23.02.1995.

24. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету / [уклад. Бойченко С.В., Іванченко О.В.]. – К. : НАУ, 2017. – 63 с.