DEVOPS

Devops means development and operational managing both works

Need to learn basics road map of devops

**Server admin** –linux - need to learn commands

-Unix

**Source code management** (SCM)

GIT/BIT bucket/GIT lab 🡪 GIT \*

**Built tool** 🡪Maven/Gradle🡪 Maven

🡪Artifactory tool

**Continous integration** (CI) 🡪 most used tool Jenkins \*

**Web servers**

🡪Apache

🡪Ngnix

🡪Tomcat

**Data base**

SQL 1) My SQL NO SQL 1) Mongo DB \*

2) Postgres SQL \* 2) Cossindra

3) MS SQL 3) Dynamo DB

**Cloud**

AWS 🡪Amazon web services\*

AZURE 🡪

GCP 🡪 Google cloud platform\*

**Configuration tool**

Ansible 🡪more used tool for configuration\*

Puppet

Chef

**Container 🡪 to make infrastructure by using this tool**

DOCKER

**Container archestars**

Kuber netis\*

Open shift

**Monitoring tool**

Gafana \*

Data dog

Nagios

**Teraform and AWS cloud formation**

**Continuous testing**

Selenium\*

**Scripting knowledge**

Python or shell scripting \*

**Learnings of devops**

Main topics of devops:

🡪Devops overview

🡪Version control using Git & GitHub

🡪Continous integration using Jenkins

🡪Containerization using Docker

🡪Kubernetes installation

🡪Configuration management

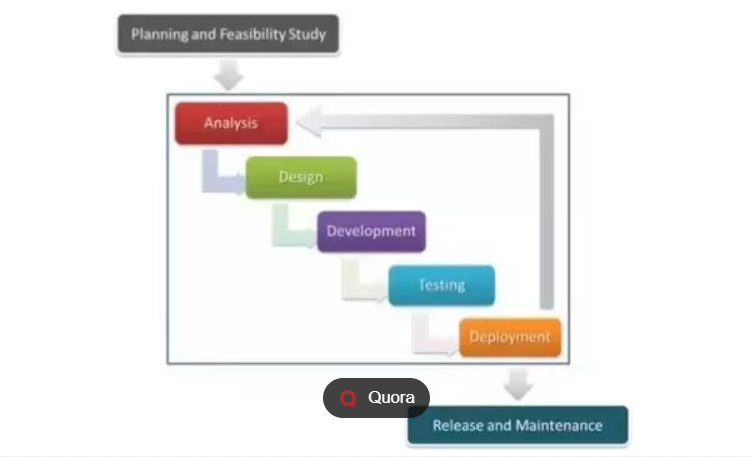
🡪Ansible

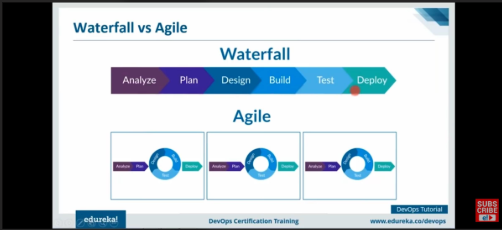
🡪Puppet

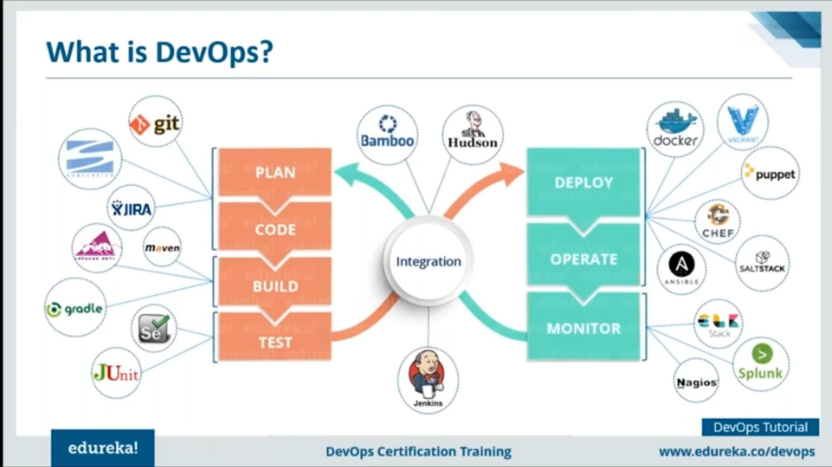
🡪Continous monitoring using Nagios

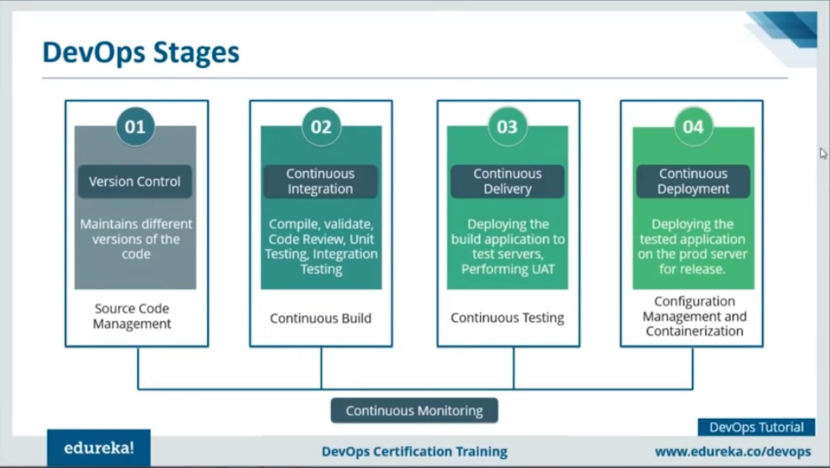
**1 -Topic:**

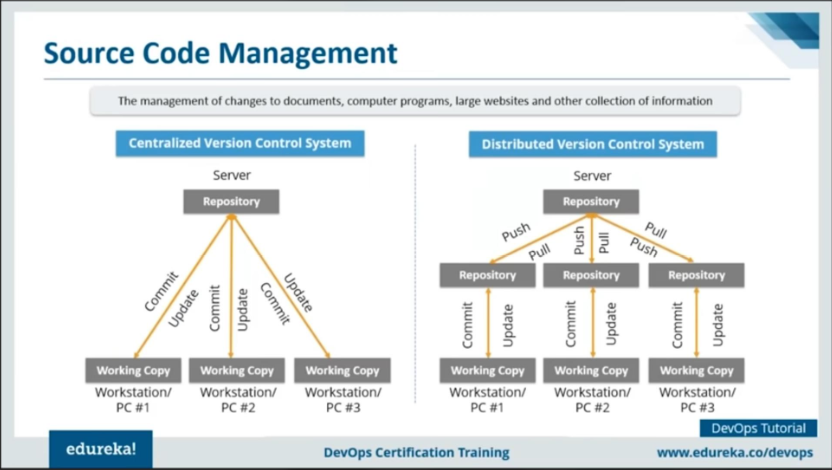
**Why devops? – Water fall** model describes a development method that is linear and sequential waterfall development step by step procedure





**What is devops? - **

**Devops stages? -** ****

****

**Devops tools?**

**Git:** Gitis a distributed revision control and source code management system with an emphasis on speed. Git was initially designed and developed by Linus Torvalds for linux kernel development. Git is a free software distributed under the terms of the GNU General Public license version 2.

Git for Project version control in a distributed environment while working on web – based and non – web based application development.

**Version Control System (VCS):** VCS is a software that helps software developers to work together and maintain a complete history of their work

Functions of VCS:

Allows developers to work simultaneously

Does not allow overwriting each other’s changes.

Maintain a history of every version

Types of VCS:

Centralized version control system (CVCS)

Distributed/Decentralized version control system (DVCS)

**Advantages of Git:**

**Free and open source**

Git is released under GPL’s open source license. It is available freely over the internet .You can use git to manage property projects without paying a single penny. As it is open source, you can download its source code and also perform changes according to your requirements.

Let us see the basic workflow git.

Step 1 – modify a file from the working directory.

Step 2 – add these files to the staging area.

Step 3 – perform commit operation that moves the files from the staging area. After push operation, it stores the changes permanently to the git repository

# First commit

[bash]$ git add sort.c

# adds file to the staging area

[bash]$ git commit –m “Added sort operation”

# Second commit

[bash]$ git add search.c

# adds file to the staging area

[bash]$ git commit –m “Added search operation”

How to install git

Download git for windows

After downloaded complete go to command prompt

Git -after enter

**Git version**

Cd<file path>command -$ cd Desktop/test/

**Git clone command**

$ git clone URL

**Git status command**

$ git status

Git add <file name> command

**Repository: Main branch 🡪git clone url**

**Sub branch🡪git-b-sub branch name url**

* What is GitHub
* Create and clone the repository
* How to create a branch
* How to commit the changes
* Create pull requests and merge
* Push code into Git repository
* Change default branch
* Delete repository
* Handling errors
* Create organization repository
* GitHub is an **hosting service platform** (hosting means store multiple files) and distribution version control
* On Git we can able to store any type of files by creating a **repository**
* GitHub encourages **teams to work together** to build and edit their site content
* **Ex:** If there is ABC persons worked on same project so there is multiple developers worked on project like home page –dashboard –about after particular they save the latest code and need to deploy in the product

Headers of git app

* Pull requests is for merge the codes or create new own pull requests we use
* Issues is for if any errors or changes mention in the issues block so we can solve here the issues
* + symbol is for creating projects or any organization

**Git commands:**

**1. Initialize a local Git repository 🡪** git init (installation)

**2. Create a local copy of a remote repository🡪** git clone ssh://git@github.com/[username]/[repository-name].git

**3. Add a file to the staging area🡪** git add [file-name.txt]

**4. Add all new and changed files to the staging area 🡪** git add –A

**5. Commit changes 🡪** git commit –m “(commit message)”

**6. Remove a file or folder 🡪** git rm –r [file name.txt]

**7. Create a new branch 🡪** git branch [branch name]

**8. Delete a branch🡪** git branch –d [branch name]

**9. Merge a branch into the active branch 🡪** git merge [branch name]

**10. Merge a branch into a target branch 🡪** git merge [source branch] [target branch]

**11. Push a branch to your remote repository 🡪** git push origin [branch name]

**12. Push changes to a remote repository 🡪** git push

**13. Pull changes from remote repository 🡪** git pull origin [branch name]

**14. Add a remote repository 🡪** git remote add origin

ssh://git@github.com/[username]/[repository-name].git

**For adding file from local to repository**

🡪git init

🡪git add README.md

🡪git add –all

🡪git commit –m “first commit”

🡪git branch –M main

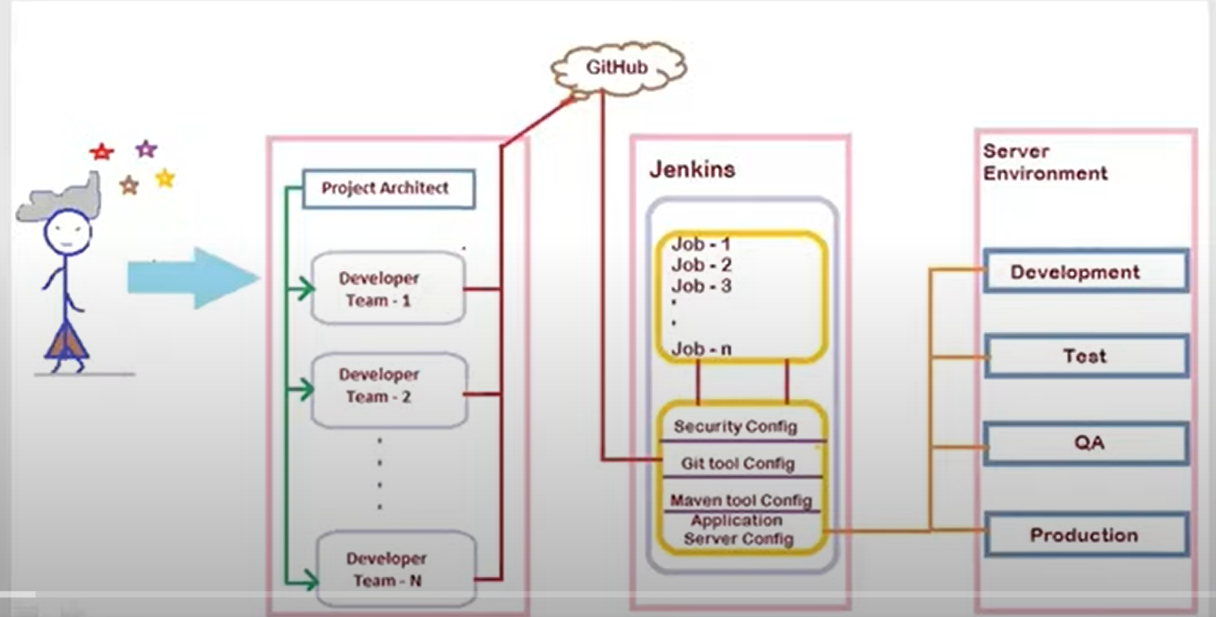
🡪git remote add origin <https://github.com/dharannuthan/begginer.git>

🡪git push –u origin main

**For clone in a local file**

🡪git clone url --take a link from repository

**Jenkins**:



**What is CI/CD?**

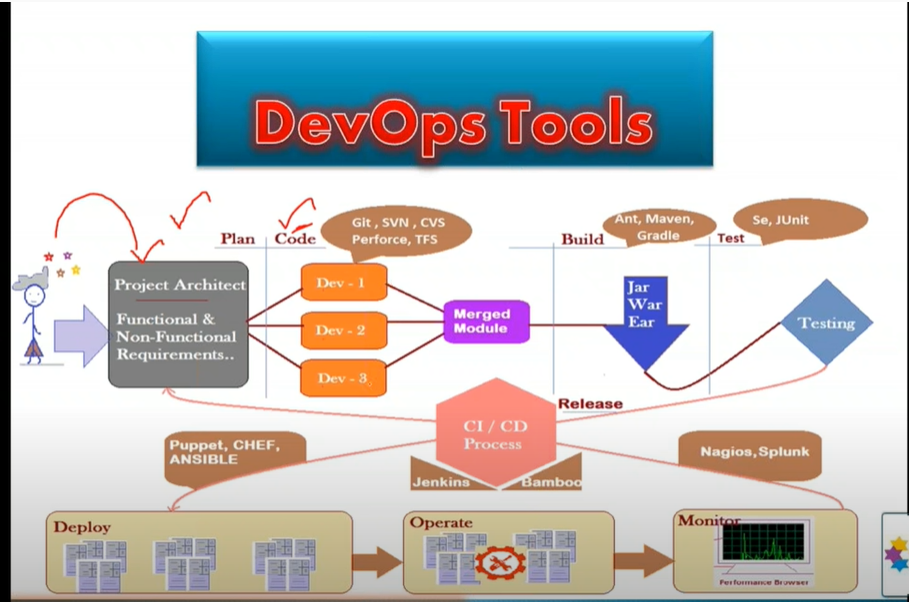
**Continuous integration:** the continuous integration is a process of merging all developers work in shared environment

**Continuous Delivery:** It is the best approach to generate product in short cycles, which is work done by developers in their shared environment

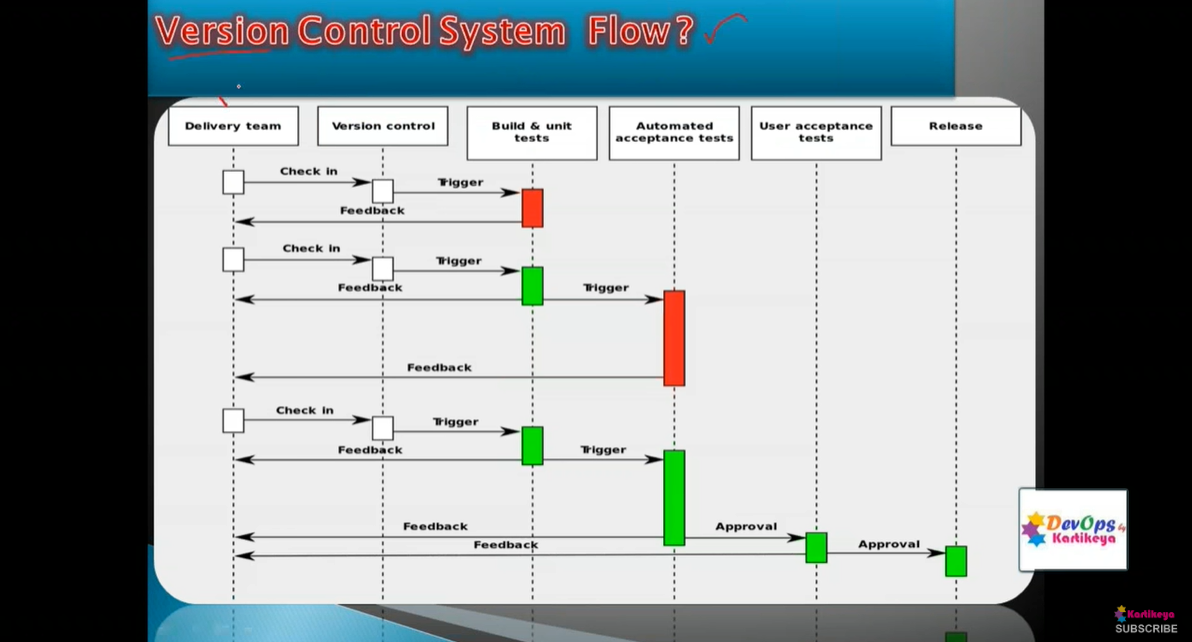
**What is plugins in Jenkins**

Plugins extend the functionalities and capabilities of Jenkins to make it suitable for all user-specific needs. These plugins encompass source code management, administration, platforms, UI/UX, building management, and much more. They help Jenkins integrate with practically every tool in the CI/CD [toolchain](https://www.opsera.io/learn/beginners-devops-toolchain).

**Git :**



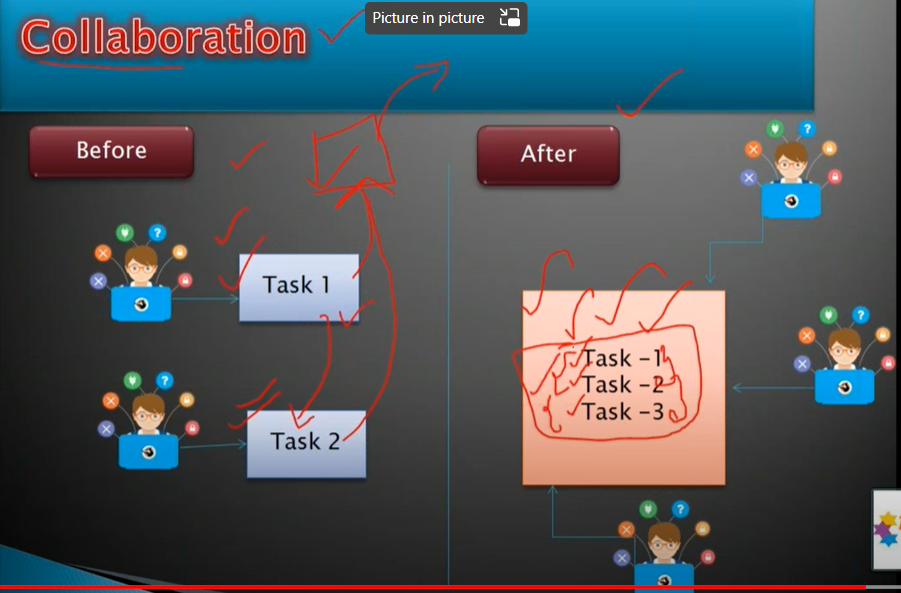
Version control system:



**VCS(Version control system):**

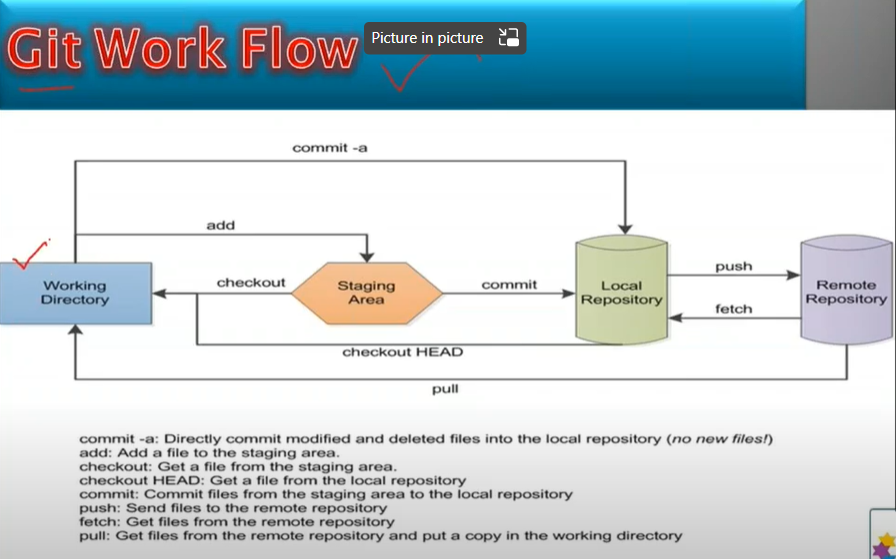
**VCS** is a process to store all members(developer) changes in centralized location and keep track all changes being done in the vcs by whom, when & why

**Collaboration:**

****

**The** tasks done by different developers stored in repository to through checking of the code to send for a build and also check the code step to step of requirements of all tasks of developers so we can find error easily

**Git Work flow:**

****

The start work is done at work directory after coding or changes they will send manually by using commands to staging area

\*If the code is exact without any errors we can move to project they can push to local repository .If any changes need to change they can revert back to modify errors

\*If there is any changes or errors we can revert back to staging area or working directory

\*If there is any changes we must mention the comment while adding back to local repository

Git:

**Git push:**

🡪**git init**

**🡪git add . (or) git add--all**

**🡪git commit –m “commit message“**

**🡪git status**

**🡪git push –u origin “branch name” ( OR ) git push Url**

**🡪git status**

**Git Clone:**

**🡪git branch**

**🡪git branch name**

**🡪git clone url**

**🡪git add –all**

**🡪git status**

**In git bash while command to any directory C:\Users\MICROGRID\Desktop\negi we should take / this one to execute**

**What is git revert**

**The git**  revert commit command is an “undo” operation that provides user with a careful, less intrusive way to undo changes

Git revert is a safe, forward moving alternative to git reset when faced with the possibility of losing work git revert removes all the changes that a single comment made to the source code repository for instance if a commit added a file called Wombat.

HTML to the repository the git can remove that file (or) if a previous commit added a line to python file, a git revert done on that previous commit will remove the offending line

**How does git revert work :**

Rather than removing the commit in question from the project history, the git revert figures out how to invert (or) reverse the changes created by the commit, then appends a new commit containing the inversed content this process stops git from losing history

**How do you revert git commit:**

To perform git revert commit, you must have the ID number of the commit in questions

**The steps to follow :**

* Use the git log or reflog command to find ID of the commit you want to revert
* Enter the git revert command, including the commit ID you want to work on
* Provide an informative git commit message to explain why you needed to perform the git revert

🡪git push

🡪git add .

🡪git commit –m “commit message”

🡪git commit –m “commit message”

🡪git push

If there is any unnecessary change

🡪git add .

🡪git commit –m ”unnecessary commit message”

🡪 git push

Before we see git revert we need to take (git log)

🡪git log

🡪:Q git revert id number🡪 need to give file path of git & repository

At last we need to git push the previous changes will be saved latest changes will be not add

**How to take meta data backup for sales force:**

Need to login in sales force account 🡪 Test.salesforce.com

**Sand Box:**  To store the application URL in sand box

We need to connect an organization

We need to select project

For some projects we need to take package.xml in Herioq application

For some projects we need to clear visual studio code and need to place package.xml and we need to clear Force stop data for every time when we take backups.

After package.xml we click on right click after we need to select **retrieve src organization** then the files will retrieve in force stop data.

**Sales force data base backup :**

Need to login in sales force after

we need to take data loader.io 🡪 New tab again we need to login salesforce

Need to click on New task 🡪Export 🡪Master configuration 🡪 Need to select all boxes options

Check the total records and save and run

**LINUX:**

Linux is a family of open source unix operating system based on the linux kernal. They include Ubuntu, Fedora, Debain, open suse and red mat.using linux to manage a virtual private server (VPS) is common practice

**Google Cloud Platform**

**What is app engine GCP?**

App engine is a fully managed, server less platform for developing and hosting web application at scale.

There are two environments Standard and flexible environment

|  |  |  |
| --- | --- | --- |
|  | STANDARD ENV | FLEXIBLE ENV |
|  | App instance run in sandbox | run with docker |
|  | Apps need to deal with rapid scaling | Apps that recieve consistent traffics |
|  | source code written in specific versions of program languages | any supported programs |
| softwares required | Python 2.7, 3.7, 3.8, 3.9 | Python |
| Java 8, 11, 17 | Java |
| Node js 10, 12, 14, 16 | Node js |
| PHP 5.5, 7.2, 7.3, 7.4 | PHP |
| Ruby 2.5, 2.6, 2.7, 3.0 | Ruby |
| Go 1.11, 1.12, 1.13, 1.14, 1.15, 1.16 | Go |
|  |  | .NET |
|  | Intended to free or at very low cost |  |
|  |  |  |
| Instance startup time | seconds | minutes |
| Maximum request timeout | depends on runtime & type of scaling | 60 minutes |
| Background threads | Yes, with restrictions | yes |
| Background processes | NO | YES |
| SSH debugging | NO | Yes |
| Scaling | Manual, Basic, Automatic | Manual, Automatic |
| Scale to zero | yes | No, minimum 1 instance |
| Modifying the runtime | NO | Yes (through Docker file) |
| Deployment time | Seconds | Minutes |
| Automatic in-place security patches | yes | Yes (excludes container image runtime) |
| pricing | based on instance hours | Based on usage of vCPU, memory and persistant disks |
|  |  |  |
|  |  | vCPU $0.06/Core hour |
|  |  | memory $0.008/gb hour |
|  |  |  |
|  |  |  |
|  |  |  |

**How to check logs in App engine**

Log-in to app engine 🡪 need to go version 🡪 then we appear a table format need to go Tools and after we need to click on logs we get logs of application

**Cloud storage**:

In cloud storage we can store any contents and we can store images and videos after we can change in URL we can deploy in codes.

In APP engine:

Application: One App per project

Services: Multiple micro services in a single application

Version: