**Water fall model:**

Documented in the year 1970 by Royce.

Water fall model describes the development linear and sequential

This model has distinct goals for each phase of development because it’s pretty similar to waterfall model



App goes to next stage only previous is completed.

Client gives requirements for app.

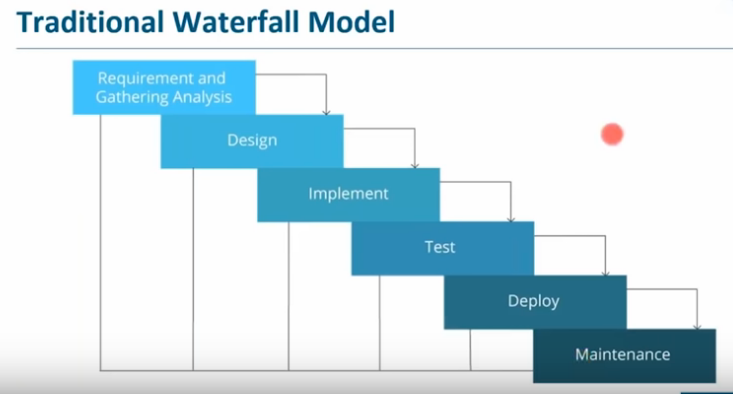
Gather that requirement and try to analyse.

Design the app how the app is going to look like.

Write code for app and we build

When we say build its build it involves multiple things, compiling your app, unit testing, it involves packaging as well,

After that it is deployed on to test, test servers for testing and then deployed on to the prod servers for release, once app is live it is monitored



Disadvantages of this:

App is in testing stage, it’s very difficult to go back and change something.

If you have written code for entire code app, in testing you found bug, in order to remove that bug you need to go to entire source code of app , that takes lot of time.

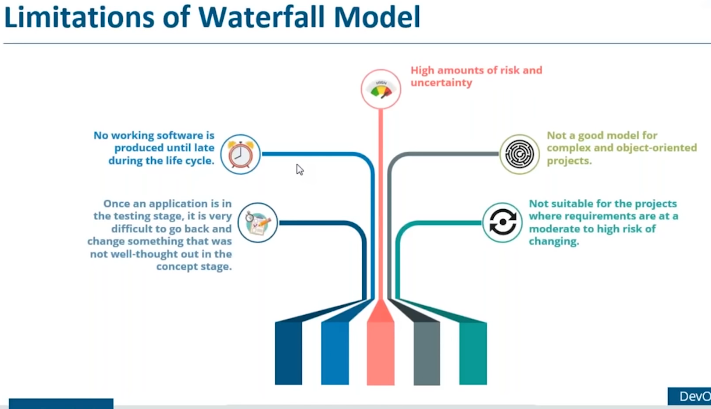
There are high of amount of risk and uncertainty ,i.e once prod is live and there is market , if there is bug or any down time, you need to go to entire source code, we need to go through entire process of waterfall model again that we saw in order to produce a working software again , and there is lot of risk and uncertainty and time consuming.

Imagine you have upgraded a software stack in prod environment that lead to failure of your app, It takes lot of time to roll back to previous process.

Not good for complex and object-oriented projects

Not good for where projects are changing to moderate to high risk of changing i.e :-

Suppose your client have given you a requirement for web app . you took your time . you’re in a condition to release your app after one year, after 1 year the market has changed , your client does not want that web app he is looking for mobile app now. This type of model is not suitable where requirements are at a moderate to high risk of changing.



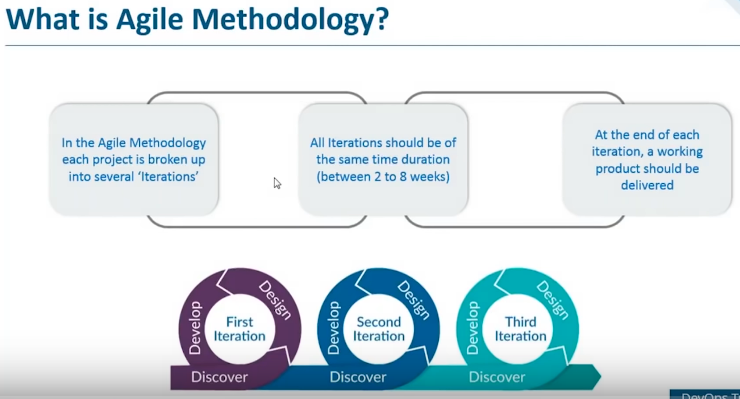
Agile Methodology:

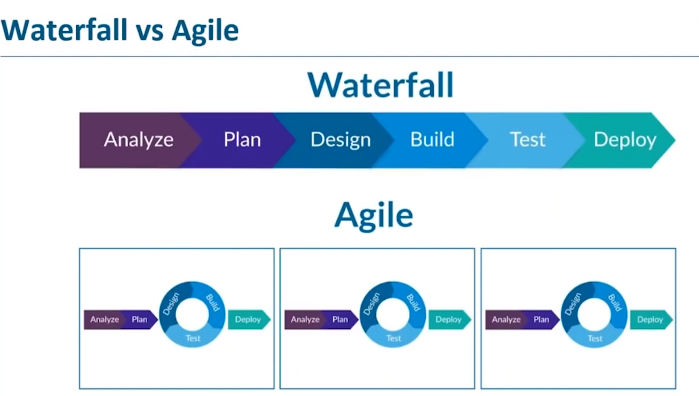
It’s a practice that promotes continuous iteration of development and testing through out the software development of the project

Development and testing used to happen continuously in agile methodology.

We can see in fig multiple iterations involved:

We designed and developed again we discovered and designed and developed





Water model is linear and pretty straight in fig

We can observe In Agile model build, design and testing part is happening continuously (we are writing the code and building it & testing it continuously and there are several iterations are involved in this stage, once final testing is done it is deployed on to the prod server for release

So Agile methodology breaks the entire life cycle into small sprins or iterations due to which development and testing part of SDLC happen continuously

Limitations of Agile :

The dev part of the team was pretty Agile.

Dev and testing happens continuously but when I talk about deployment that was not continuous

There was still lots of conflicts happening b/w dev & ops side of company

The dev team wants agility and operation team want stability

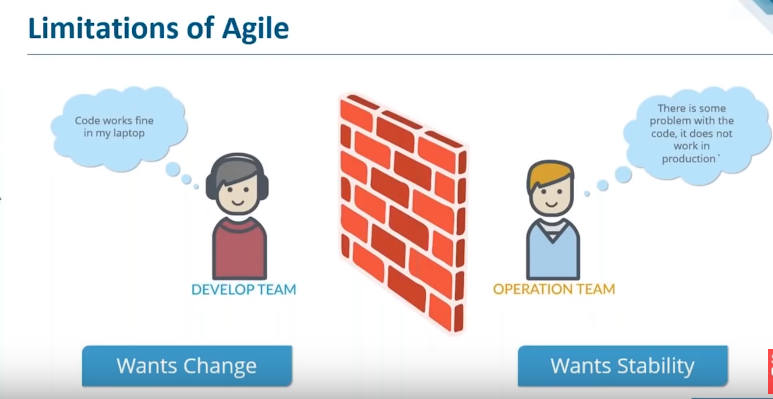
Code works good in dev lap, when it reaches to production, there is an bug in app or it does not work in prod at all ,this is bcoz there is an inconsistency in the computing environment that has caused that

Due to which dev team and the operation team used to fight a lot, lot of conflicts at that time

Here deployment is still linear, you need to deploy it manually on to the various prod servers that is happening in Agile methodology

Once your app is live due to software stack in the prod environment and it does not work properly, now to go back and change in the prod environment used to take a lot of time.

Ex : Upgraded some software stack due to that u r app is not working , now to go back to the previous stable version of software stack , the operations team was taking lot of time because they have to go through the long scripts that they have written in order to provision the infrastructure.



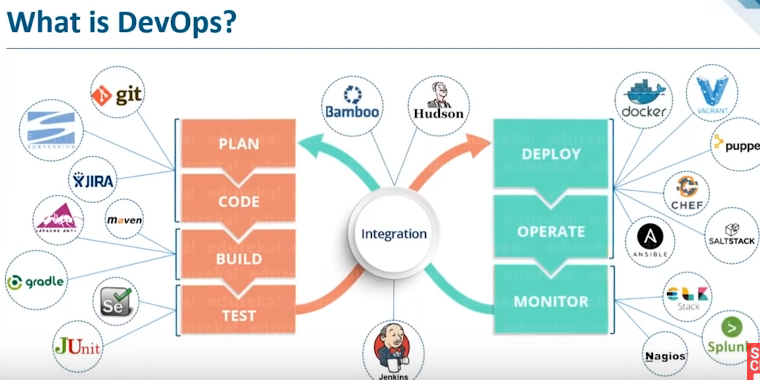
Solutions to all other problems that we discussed:

Devops Is basically a software development strategy which Bridges the gap blw dev side and ops side of company, it’s a basically a term for a group of concepts that while not all new, have catalyse in to movement and a rapidly spreading throughout the technical community.

Its not a technology and it’s a methodology, basically it’s a practice that is dedicated to study building, evolving and operating rapidly changing the systems at scale.

Devops is the practice of operations and development Engineers participating together in the entire life cycle (SDLC) from design through the development process to production support.

Devops is also characterized by operation staff making use many of the same techniques as developers for their systems work.

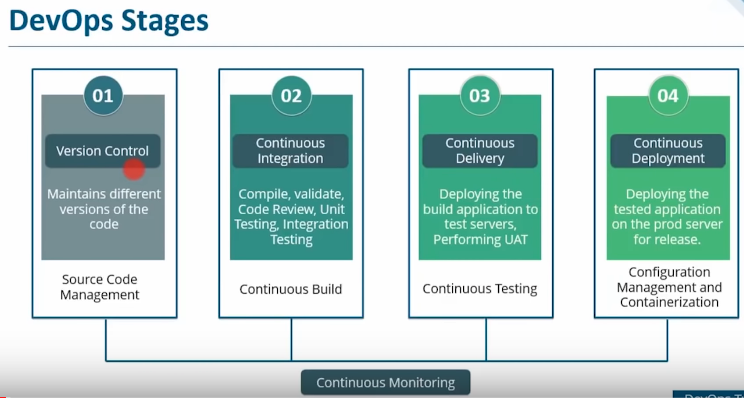


Devops Is basically a software development strategy which Bridges the gap blw dev side and ops side of company and helps us to deliver good quality software intime.

Every things happens continuously here

How can i use these tools.

Various stages involved in devops:



Version control :-

How I know which developer has made which commit and at what time code, and which cimmit is actually causing error and how will revert back to previous code , how can I manage source code.

Continuous integeration :

Basically building your app continuously , any dev made a changes to source code continuous integration server should be able to pull that code and prepare a build ,when I say build its not only compling and it is( compile , validate, code reviw,unit testing, integration testing and packaging your app aswell )

Continous Delivery :

Suppose Jenkins – what Jenkins do is, once app is build , it will be deployed on to test servers for testing performance , user acceptance test ,or end user test there we are using selenium for performing automation testing ,once that is done deployed on to the prod servers for release that is called continuous deployment

Here we will be using configuration management and containerization tools : Here we are provisioning your infrastructure to provison your prod environment

Contiinous deployment is not a good practice to that dat , before releasing your product in to the market there might be multiple checks that you want to do before that ,there might be multiple other testings you want to do ,so you don’t want this to be automated

so after continuous delivery we can go ahead and use manual configuration tools such as Ansible puppet ,chef we can also use Docker for similar purpose then we can go and deploy on to the prod service , once app is live it is continuously monitored by NAgos or spluk which provides the relevant feed back to concerned team.

What Jenkins do ?

The moment when developpers do changes in Source code , it takes that code and trigger a build using tools like maven or ant or gradle once that is done it will deplouy on to the test servers for testing for end user testing using tools like selenium , Junit etc. then it will automatically takes the tested app and deploy on to the prod servers for release ,then it is continuously monitored by nagos , splunk Elk etc .

Jenkins is an heart of Devops life cycle.

2 types of source code management

* **Centralized version control** : uses centralized server to store all the files and enables team collaboration , its works in an single repository to which users can directly access a central server.

If any one committed the code we will have commit id , and who committed and at what time code had commited

Disadvantages:

Network should be available

If any centralized server crash , all your code is last

* **Distributed version control system :**

Here no need to rely on central repository , every contributer has a local copy or clone of the main repository . every programmer contain local repository on its own which is actually a copy or clone of the central repository on their hard drive , they can commit and update local repository without any interference . they can update their local repositories with the new data coming from the central servers by an operation called pull and affect chanfes to the main repository by an operation called push from local.

Advant:

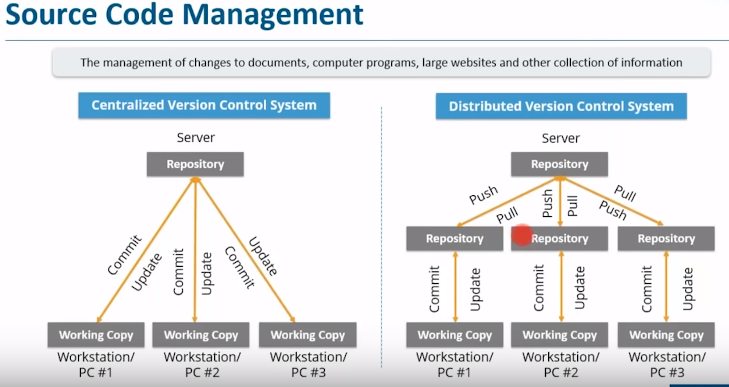
No need of internet connection

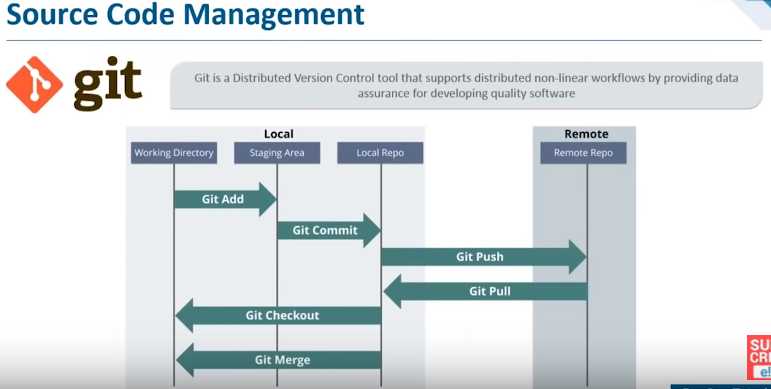
Committing new change sets can be done locallyWithout manipulating the changes on the main repository , once you have all the group sets ready you can push all at once. We can review once and we can push .

If the central servers gets crashed we can easily recover from any one of the contributor local repository

They can share changes with one another if they want to get the feedback , before affecting the changes in the main repository aswell.

So this is better.





Git - -version

Git init : initialize empty directory as local git repository

Touch edureka.py

Gedit edureka.py -🡪to edit a file

Git add edureka.py 🡪add files to the staging area

Git status 🡪tells files need to be commmited

Git commit “first msg”

Repository -click add -givename-create repository

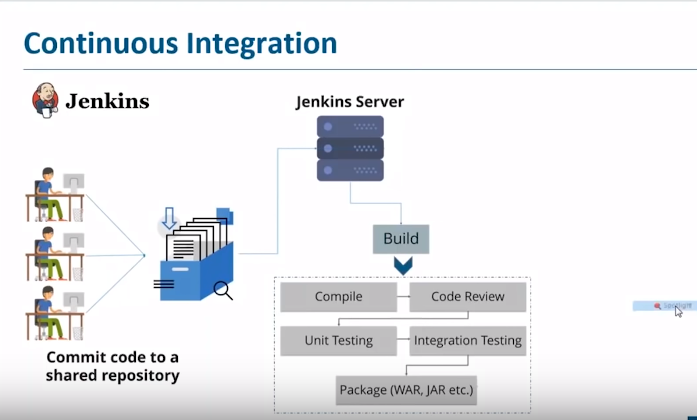
Git remote add origin “ paste that clone link here” -added my remote repository to mylocal repository

Git pull origin master -pull 🡪whatever is there in my remote repository to local git repository is pulled

Git push origin master

Containers integration:

It is basically a development practice in which developers are required to commit changes in source code in shared repository more frequently or daily . Every commit made in the repository is build ,This helps the teams to detects the problems very early.



Here you can see many developpers are committing the co de to the shared repository , from there Jenkins server i.e : containers integration tool will pull that code the moment any developer commits the change in the source code in source code, Jenkins servers pull that & it will prepare a build.

Build doesn’t only mean compling it includes compling and aswell as other things such as code review ,unit testing etc…,integration testing ,package (WAR,Jar etc). it happens in a container

the moment any developer commits the source code, Jenkins pull that & prepare build , ths is called containers integration

Jenkins has various tools to do this

It has various tools for developing, testing, deployment technology and it has well over 2500 plugins and we can goahead and trigger the whatever the job you want

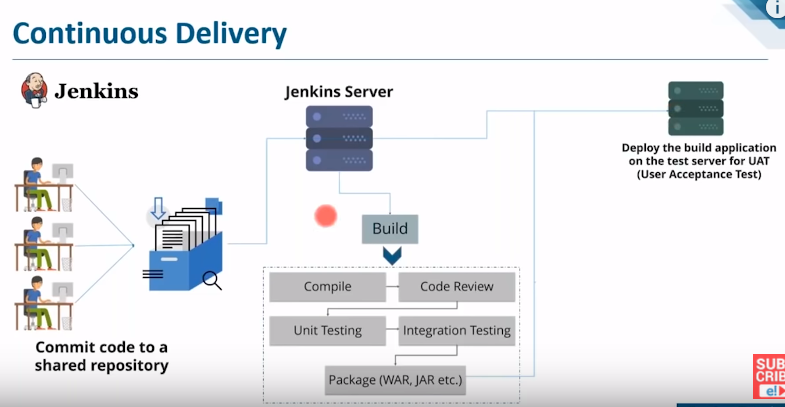
**Containers delivery**: It means taking containers integration to the next step

In a container’s manner or in an automated fashion we are taking the build app on to the test server for end-user testing or unit or user acceptance testing this happens in a containers fashion.

the moment any developer commits the source code, Jenkins pull that & prepare build once build is successful Jenkins takes that build app and Jenkins will deploy on to the test server for End user testing & user acceptance testing

The package that we have created here i.e jar or war or executable file Jenkins takes that package and it will deploy on to the test server for End user testing this kind of testing is called end-user testing or user acceptance testing where u need to deploy your app on to the server which can be replica of prod server

For ex :if I want to check functionality test of my app,I will check whether search engine is working or not,people are logging or not, all the f/n’s of website is checked basically after deploying on to the server that’s called f/ncal testing



The advantage here is when there is a build failure we can know which commit has caused that error ,instead of going through the entire source code of app.

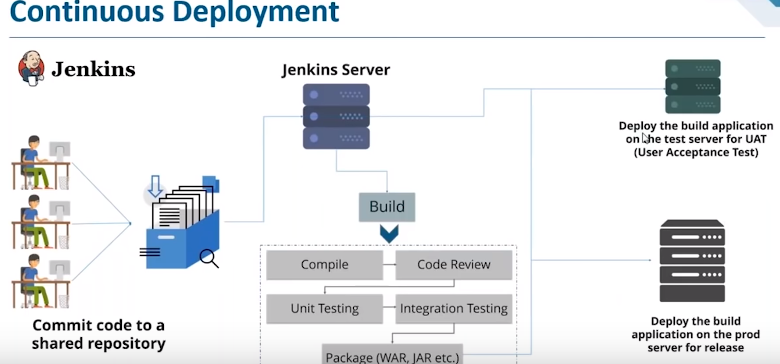
Similarly in testing if any bug appears in the testing is well we know which commit has caused that error, we can go ahead ,we can have a look at that particular commit instead of going through entire source codes

Systems allows us to detect the problems early.

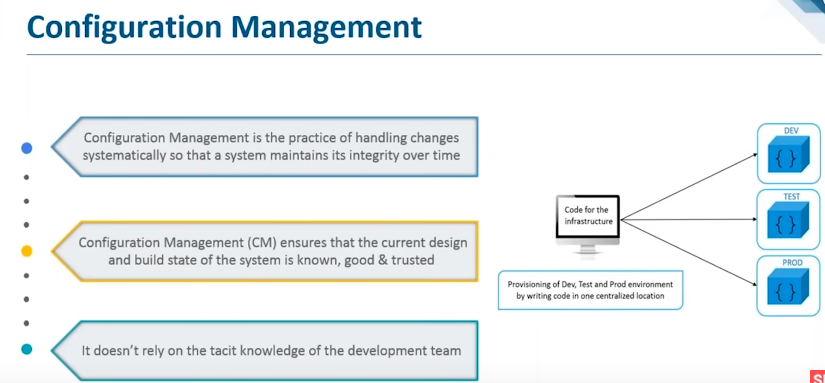
**Containers deployment**:

It is basically taking the build app that you have tested and deploying on to the prod servers in automated fashion. once app is tested it is deployed on to prod servers in automatically.

Once app is tested then deployed on to the prod servers for release and its not good to deploy your app continuously or in an automated fashion. Bcoz you may want to market that app.



**Configuration Management :**



Few issues with deployment of app or provisioning of server:

I build my app when I deploy my app on to dev or prod servers There can have some dependency issues so app not work, ex : dev lap some software stack updated in prod environ they are still using outdated version of that software stack.

ex2 : when app is live and it goes down bcoz of some reasons like we have updated some software stack then how u rollback to previous stable version of this software stack.

So there are lots of issues in the admin side of the company and operation side of company that can be removed using config management tools.

Lot of problems used to have from admin side and op side of company there are eliminated usng conf management tools

Admin used to write longscripts in order to provision infrastructure whether its test environ or prod or dev environ which were prone to error to which is diificult to debug and other persons difficult to recognise that bug.

Suppose if I want to install lamstack on dev, test, prod, I write a code for installing lamstack on 1 central location and I can go ahead and deploy on to the test,dev,and prod

So I have record of this system state present in my central location

I still have the previous version of the software stack.

I don’t have to write scripts manually and deploy to the nodes.

Challenges of CM:

It helps us to figure out which components to change when requirement change

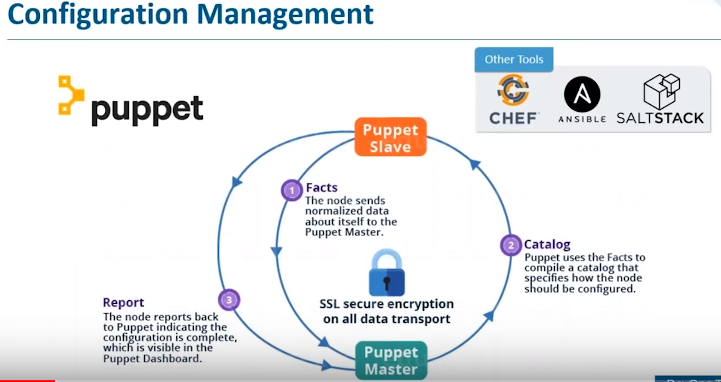
It also helps us re doing a n implementation bcoz requirement changed since last implementation

Helps us to revert to previous version of component if u have replaced with new but the flawed version

**Importance of config management: -**

**VARIOUS TOOLS AVAILABLE FOR CONFIG MANAGEMENT:-**

* **PUPPET: Deploying configuring and managing service**
  + **We can define distinct configurations for each & every host and continuously check and confirm whether the required configuration is placed and is not altered on the host . distinct configurations (distinct configurations for ex for 1 node I need this software stack and for other we need another software stack I can define distinct configs for diff hosts) and if it is altered puppet will revert back to the required configuration**
  + **It also helps us in dynamic scaling up and dynamic scaling down of machines.(ex: big billing days – we use puppet to provision more servers)**
  + **It follows master slave architecture in which the slaves polls the central server for changes made in config, we have multiple nodes which are connected in the master, they will poll , they checks continuously is there any changes happened in configuration to the master, the moment any change happened ,it will pull that configuration and deploy on to that particular node. This is pull configuration (pUPpet and chef follows pull configuration, the nodes will pull that configuration )**
  + **In push config the master will actually push the configurations ON TO THE NODES. WHICH HAPPENS IN ANSIBLE , AND SALTACK (THEY FOLLOWS PUSH CONFIG**
* **CHEF**
* **ANSIBLE**
* **SALTSTACK**



You can see the typical puppet architecture.

1st puppet agent sends the facts to the puppet master ,facts are keyvalue data pairs,aspects of slave states such as ip address, whether it is a vm ,up time,os

Puppet master uses a facts to compile a catlaog that defines how a slave should be configured.

Catalog is a doc describes a desired state for each resource that puppet master manages on a puppet slave, then puppet slave reports backs to the master indicating that config is complete which Is also visible in puppet dashboard

Containerization:

* These are Light weight alternatives to vm’s
* We have docker container that contains lib and bin required for that app. ( we have containerized a particular app)
* Here we have host os and on top of which we have docker engine, there is no guest os
* We are learning 2 containers , 1st having app1 and its bins/libs &2nd having app2 and its bins/libs
* All I need to run the app is that container bcoz all the dependencies are already present in that particular container.contaner contains my app, depencies of my app, bin/libs required for my app is there in my container

Use case of docker in industry

Suppose we have complex requirement for microservice,

We have written docker file ,with docker file we can create docker image ,its nothing but the template for u r docker container, with docker image we can create as many docker containers required

The Docker image I can upload to the docker hub which is nothing but git repository of docker image. we can have public repositories ,private repositories there , from docker hub any team staging a production can pull that particular image and prepare as many containers as they want .

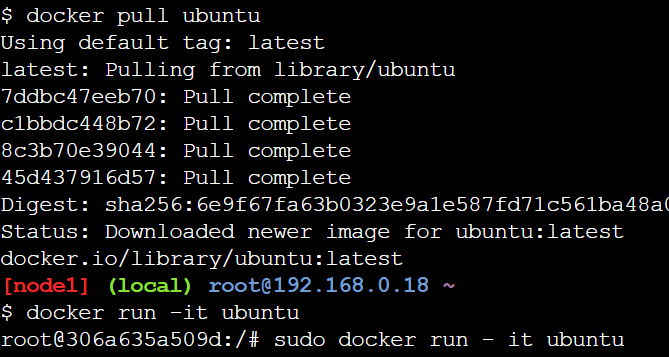
Whatever is there in my developers lap, the microservices app guy who has written that and the requirement for microservices app , that guy is basically a developer and he is only developpping the app,whatever is there in my developer lap I have replicated in my staging and prod, this is consistent computing envrinment throughout software development life cycle

[briefly :visualize containers is an box where u r app is present along with all its dependencies except the box is infinitly replicable, whatever happens in the box, stage in the box unless you explicitly take something out and put something in and when it breaks you throw it away and get a new one.

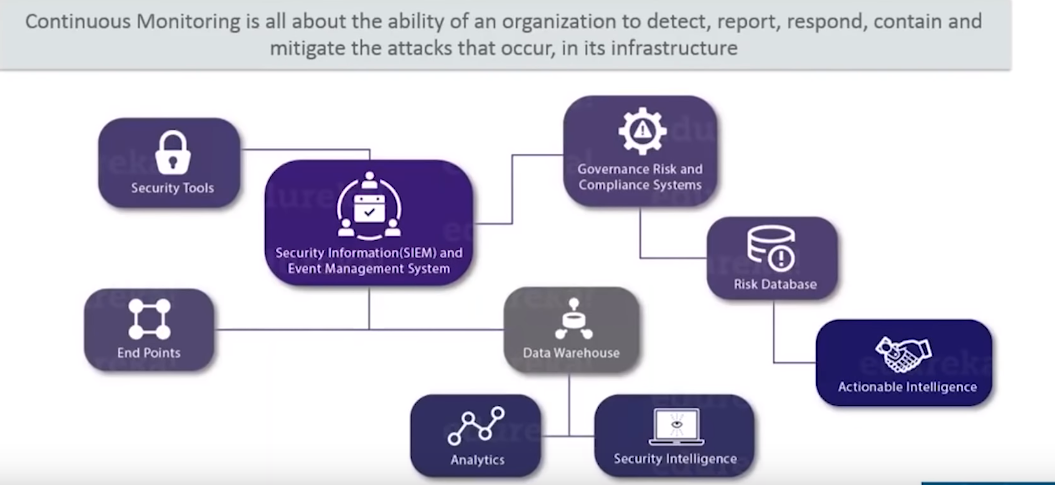
Containers make u r containers easy to run on different computers. Ideally the same image should be used to run a container in every environment stage from development to production that;s what the docker containers are.

Create an a/c on docker hub

* # Systemctl start docker
* # docker images (to display all docker images)
* # docker pull ubuntu
* # docker run-it ubuntu
* Exit



**Continous Monitoring:**



Resolves any syteem errors like low memory & unreachable server etc before they have -ve impact on your business productivity

It detects any n/w or servers problems

It determines the root cause of the any issue

It maintains th security and availability of the services and monitors in troubleshoot server performance issues.

















It is an integration of an organization security tools the aggregation normalization and correlation of data produced by security tools







**(starting from here collected from documentation)**

**Why We Need Continuous Monitoring?**

Continuous Monitoring Tools resolve any system errors ( low memory, unreachable server etc. ) before they have any negative impact on your business productivity.

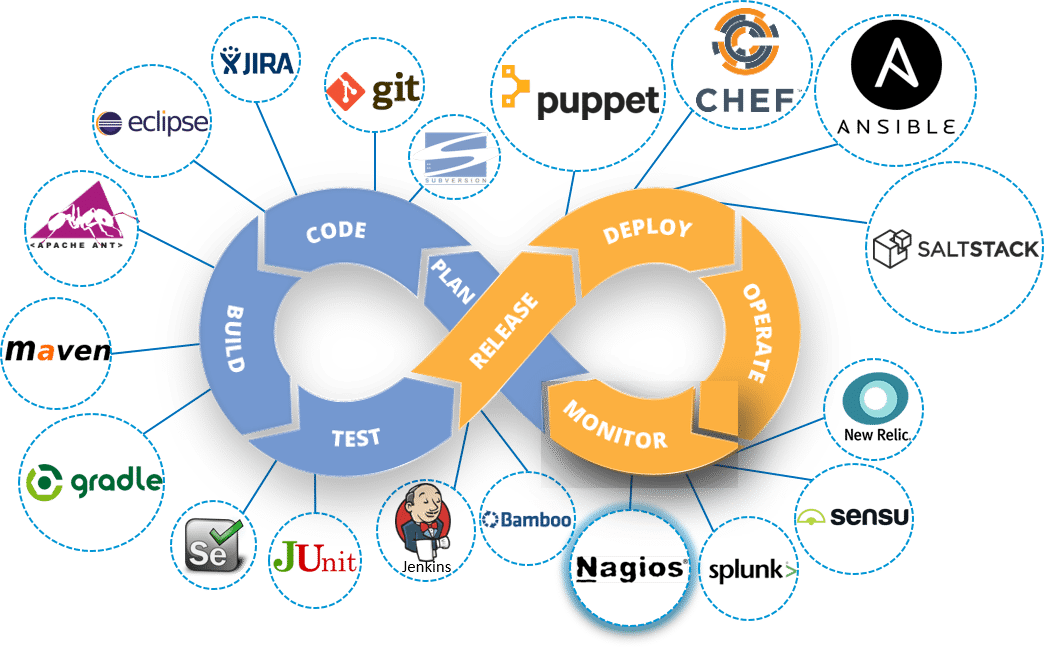
Important reasons to use a monitoring tool are:

* It detects any network or server problems
* It determines the root cause of any issues
* It maintains the security and availability of the service
* It monitors and troubleshoot server performance issues
* It allows us to plan for infrastructure upgrades before outdated systems cause failures
* It can respond to issues at the first sign of a problem
* It can be used to automatically fix problems when they are detected
* It ensures IT infrastructure outages have a minimal effect on your organization’s bottom line
* It can monitor your entire infrastructure and business processes

Yeah, it does a lot of cool work, but what is it?

**What is Continuous Monitoring?**

Let me first tell you where Continuous Monitoring lies in the DevOps life-cycle, consider the diagram below:



There are many continuos monitoring tools in the market we focous on Nagios

Nagios : it is used for continuous monitoring of systems application services and business processes in a devops culture, and incase of failure it alerts technical staff of the prob allowing the remeditation process before outage affect business processes end users or customers