Before DevOps there’s two methods exit Waterfall method & Agile Model

A screenshot of a social media post with a street sign

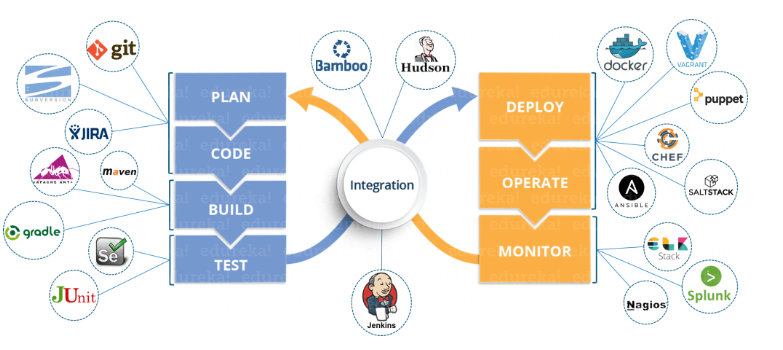
Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated



A screenshot of a cell phone

Description automatically generated

Git: Add mail & user name to git.

Git config –global user.email “[buddaprem93@gmail.com](mailto:buddaprem93@gmail.com)”

Git config –global user.name”budda”

**Create a new local repository : git init {like work station on your local system}**

To see the Hidden file files:

defaults write com.apple.finder AppleShowAllFiles -boolean true ; killall Finder

Add file: touch file name

Add ,commit

Git add file name. {git add \*.\* ‘for all file to add’ }

Git commit -m”msg”

Git status

Edit file: vi file name / view file : cat file name

Connect to Server :

Log in to git Hub account

Create repository

Get URL

Open terminal: git remote add origin URL

git remote add githubRepo [git@github.com:buddaprem/html.git](mailto:git@github.com:buddaprem/html.git)

git push -u githubRepo master

Push: git push -u origin master

Pull: git pull

Search: git grep “foo()”

Create a new branch and switch to it: git checkout -b < branch name >

Switch to branch: git checkout < branch name >

List all breach: git branch

Delete Branch: bit branch -d < branch name >

https://confluence.atlassian.com/bitbucketserver/basic-git-commands-776639767.html

MAVEN:

Is for build tool

Files in JAR/WAR/Metal

Depedencies

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A picture containing screenshot

Description automatically generated

A close up of a keyboard

Description automatically generated

DevOps: complete automation

Have 5 Phases for every aspect of complete task or release product

1.Development (for writing program). git, Maven

git, Maven

git: Source code -> store (save & version control or share repository)-> merge all code

Maven: build tool: for every project there is 1000’s of Java files from different developer -> convert into Single file is called WAR file.

This WAR file needs to be deployed ’release’ with help of Apache

Web application resource.

2.testing.

After test complete, if there’s any bugs they return to developer to fix them

Then live servers (whatever customer using like apple or Amazon……) or production server

3.Integration.

Jenkins & bamboo & team city.

checking your code several times a day and testing push that code to share repository ‘save code for backup’. That’s way you can detect problems

Once the programmer write code every day, we need to push that code(store) and Test that code So you can fix bug’s same day.

Easy for everyone programmer and testing.

Continuous testing:

4.delivery & deployment. Release updated code to Customer’s (10000s of servers)

Ansible & puppet & Chef.

5.Monitoring: feedback from your applications from customers or monitoring tools

Nagios new Relic Zabbix

Like your sever is slow, memory is low

Java: high level language

C: assembly to interact with Machine

Python: script language

Docker:

Docker images –help

Docker pull image (name like ubuntu, Jenkins, http,,,,,): docker pull Jenkins.

Docker image: To see all all available images

Docker image -q : sort with id

RUN: Docker run Jenkins (once you run image container will create)

Container details: Docker ps -a

Run Container: Docker run -it image name

Give name to container: docker run –name ubuntu

Delete: Docker rmi image name / docker rmi -f image name

Container list : docker ps /docker ps-a

A screenshot of a cell phone

Description automatically generated

Client pull request Demon so he check is there any available image if not he send request to Registry.

Inside Container : ls

Start : to Specific container RUN: Docker start/stop Container Name/ID

Pause: Docker Pause/Unpause Container Name/ID

TOP: Docker top Container Name/ID {Detilas}

Stats: Docker stats Container Name/ID {Memory CPU details}

Kill: docker kill name/id

History:

Docker file: is a text file where we give/kept instructions after we build it becomes An Image.

Syntax

Step1: create a file named Docker file.

Step2: FROM ubuntu /scratch ‘empty image’

Maintainer budda <prembudda >

RUN apt-get update

CMD [“echo ”, “this is my first image”]

Step3: Docker build location of file / “.”

Go to folder location -> docker build .

Docker image id

Docker run image id

Docker Compose: file name end with .yml

Is for run multi applications at a time

Syntax:

Version: ‘3.0 ‘

Service

Web:

Image: nginx

Database:

Image: redis

Run: docker-compose up -d

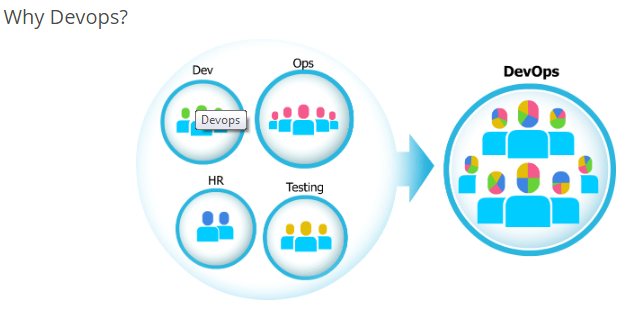
Available containers: docker ps

Stop: docker-compose down

Scale: docker-compose up -d –scale database=4 “ it will create 4 instances “.

**DevOps** is a software development approach which involves Continuous Development, Continuous Testing, Continuous Integration, Continuous Deployment and Continuous Monitoring of the software throughout its development life cycle. These activities are possible only in DevOps, not Agile or waterfall, and this is why Facebook and other top companies have chosen DevOps as the way forward for their business goals. DevOps is the preferred approach to develop high quality software in shorter development cycles which results in greater customer satisfaction.

A picture containing electronics

Description automatically generated

**What Is Continuous Deployment?**

This is the stage in the DevOps life cycle where the Software is developed continuously. Continuous Deployment is a software development practice where all code changes, whether they are new features or bug fixes, are automatically built and pushed to multiple staging environments - including production. It's similar to Continuous Delivery but doesn't rely on someone manually green-lighting and pushing code changes to the servers.

Coding and Building: Git and SVN for maintaining the different versions of the code.

Building/ packaging: Ant, Maven, Gradle for that code into an executable file that can forwarded to the QAs for testing.

**Pros:**   
The major advantage of using Continuous Deployment is that it provides greater reliability and a more efficient workflow - a key advantage when you're working with a 'move fast and break things' team mentality. Using Continuous Deployment tools to automate tasks like code minifying, asset building and review that historically required a lot more manual involvement frees up the team to continue coding, saving time and resources. And should a deployment prove to be problematic, these deployment tools make it much easier to roll back to a stable previous version.

**Cons:**

Of course, bugs happen or maybe you work with a manager who prefers a more old-school, 'check and double-check everything before it's shipped' approach. Not that long ago, most software companies released product updates on a yearly basis, giving development and QA teams plenty of time to test, fix, and test code again before shipping to users.

**Continuous Testing:**

This is the stage where the developed software is continuously tested for bugs.

Selenium TestNG, JUnit, etc automation testing

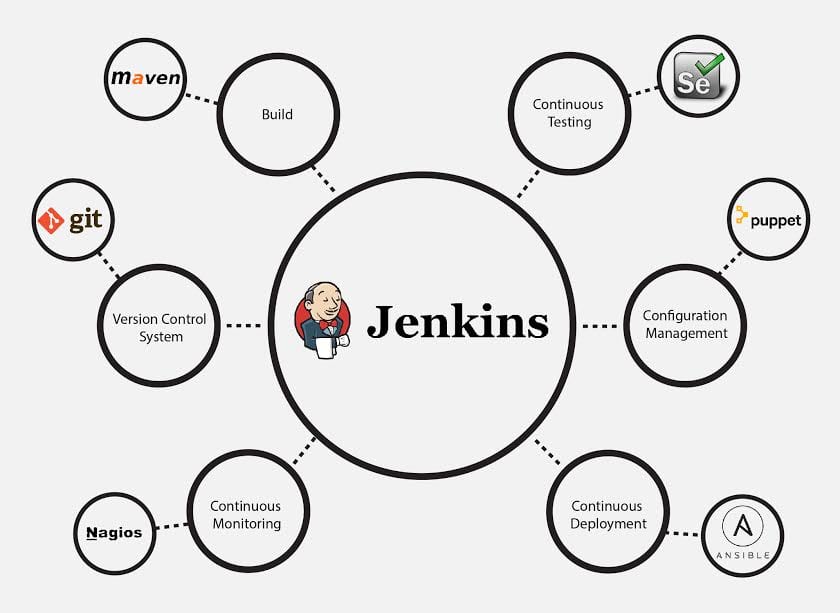
These tools allow the QAs to test multiple code-bases thoroughly in parallel to ensure that there are no flaws in the functionality. **Docker** containers for simulating ‘test environment’ on the fly, is also a preferred choice. Once the code is tested, it is continuously integrated with the existing code.

**Continuous Integration:**

The idea behind Continuous Integration is to further increase speed to market by automating the testing process after each code change.

Usually the tests are ***separated in two parts***, the UI testing and the unit testing. The UI, or user interface, testing focuses more on how things should work from a user's perspective. Does the button work? Does the content make sense? Are all the links active? Unit testing focuses more on low level code testing that is necessary for developers. Unit testing is typically run after each code change to ensure everything works as it should.

**Jenkins**: Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies. If you want to integrate a particular tool, you need to install the plugins for that tool. For example: Git, Maven 2 project, Amazon EC2, HTML publisher etc.

The image below depicts that Jenkins is integrating various DevOps stages:

**Advantages** of Jenkins include:

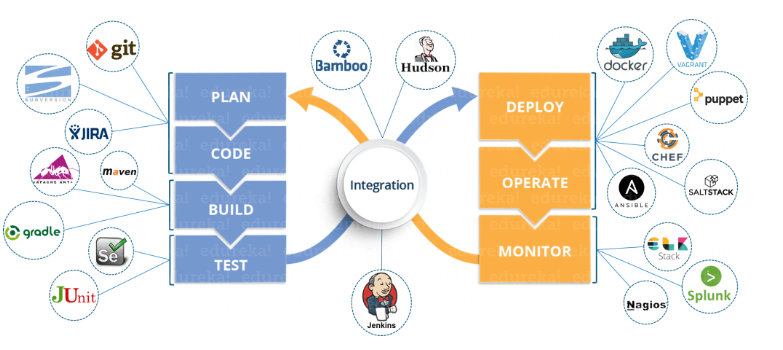
* It is an open source tool with great community support.
* It is easy to install.
* It has 1000+ plugins to ease your work. If a plugin does not exist, you can code it and share with the community.
* It is free of cost.
* It is built with Java and hence, it is portable to all the major platforms.

**Continuous Deployment:**

It is the stage where the code is deployed to the production environment. Here we ensure that the code is correctly deployed on all the servers. If there is any addition of functionality or a new feature is introduced, then one should be ready to welcome greater website traffic. So it is also the responsibility of the Sys Admin to scale up the servers to host more users.

Since the new code is deployed on a continuous basis, configuration management tools play an important role for executing tasks quickly and frequently. Puppet, Chef, SaltStack and Ansible are some popular tools that are used in this stage.

Containerization tools also play an important role in the deployment stage. Docker and Vagrant are the popular tools which help produce consistency across Development, Test, Staging and Production environments. Besides this, they also help in scaling-up and scaling-down of instances easily.



A screenshot of a cell phone

Description automatically generated

**MAVEN:**

mvn -version

mvn archetype: generate (Download a lot of files for First Time Only)

\*sensitive contains): 1388: 1388

Choose org.apache.maven.archetypes:maven-archetype-quickstart version:

1: 1.0-alpha-1

2: 1.0-alpha-2

3: 1.0-alpha-3

4: 1.0-alpha-4

5: 1.0

6: 1.1

7: 1.3

8: 1.4

Choose a number: 8: 8

Define value for property 'groupId': “company name”

Define value for property 'artifactId': “any name or id ”

Define value for property 'version' 1.0-SNAPSHOT: : 1.0.0.0

Define value for property 'package' : “optional”

mvn compile

mvn test-compile

MVN test

Mvn package

mvn integration-test

mvn verify

mvn clean: for delete unassay files.

mvn site: for web server you can see on web”. & create documentation

cd target: you see site.

cd site “ls: all html files”

Now install Tomcat

* Mvn package
* Mvn clean
* Mvn clean compile
* Mvn clean test -compile test

**CHEF**

Its configuration tool (to maintain number devices or systems) like install develop delete do all things by remote control. Dealing with webservers and dB servers and load balancer servers every day.

We have two ways to install application

1. Manual “takes lot of time”
2. Script
3. Automation tool “chef”

Configuration tools:

* Puppet: pull based
* chef: pull
* Ansible: push

**chef and puppe**t are pull based configuration management tools these are client-server based.

Each and every chef client is going to request automation data from the chef server and then apply it to the client server.

**Ansible server push base** is going to login to each server and push

Example: Facebook

Adv: reduce management complexity & time

Resource: is piece of system and its desired state.

File: create / delete.

User: create / delete.

Package: install / not.

**Managing complexity:**

Organization: its independent.no sharing information

Environment: related to your work-flow.

* development env
* testing env
* staging env
* production env

**Node:** control by chef represents the servers in your infrastructure, it could be physical server or virtual server. Also be network hardware, switches, routers, etc.…

**Chef-Client node**: default is going to ask Chef server every 30 mins, to what code it should be apply in its own server to achieve desired state.

Time period is depending 30 min, 1 hr 1 day ,,,,,

**OHAI:** Discovery tool

**Receipt:** is collection of resources files, app(.rb files ruby language )

**Cookbooks:** collection of recipes and templates and cutom resources.

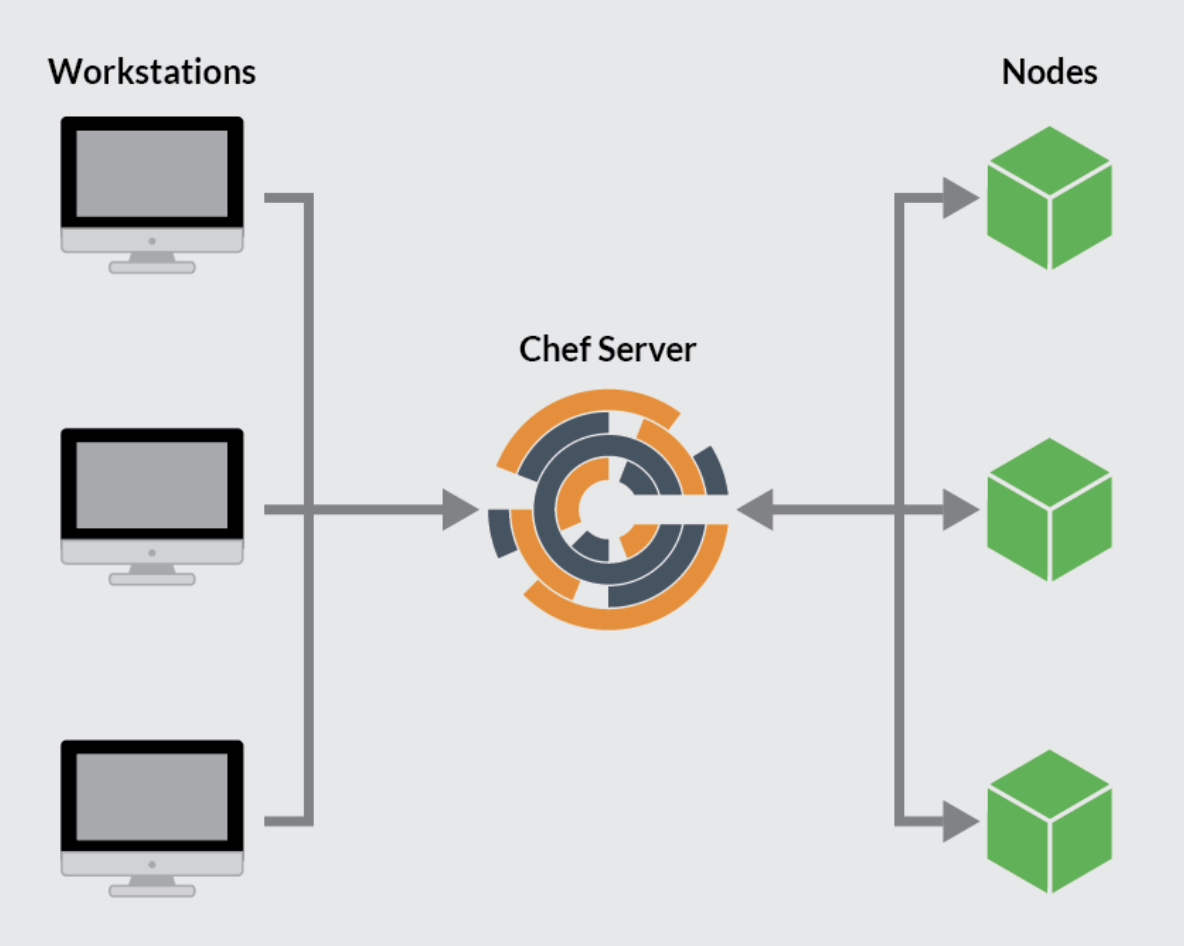
It contains all the recipes, files, templates, libraries etc.

1:1 map one cook book for one AUTO machine.

One for dB server App server network server

Policy:

**Run list:**



Work Station: where code we need to write, new features updates, bug fixing.

**No stop & no start.**

After we finish writing code we need to upload **‘push ’** to server .

Directly download & install from internet: **rpm -ivh link address**

* Chef –version
* Chef- verify

**Communication with Server**

**Upload into Server:** only throw **WORKSTATION “**no upload option on chef**.**

**Go to Administration 🡪 Organization 🡪 Starter kit 🡪**

**\*\*\*\* Make sure before Download starter kit (\*\* ask someone where starter is kit \*\*)** bcz each time you download. u get new one & old one is reset.

**Creating chef-server**

1. **Login to AWS EC2 and lunch instance with ‘chef server’**
2. **Go to Elastic IPs generate one and Associate to ‘chef server’. Its only for server no need for workstation & client advantage is ip address is same.**
3. **Go to** [**https://downloads.chef.io**](https://downloads.chef.io) **for package go to server > RHL server copy URL**
4. **Brew install rpm**

**rpm -ivh** chef-server-core-13.0.17-1.sles12.x86\_64.rpm

**creating user :** chef-server-ctl user -create chefadmin chef Admin chef@gmail.com chefadmin --filename

**create organization: chef-server-ctl org-create “flipkart pvt ltd” – association\_user chefadmin –filename flipkart-validator.pem**

once you download kit move to workstation server ‘copy’

scp-i“chef-id”/c/downloads/chef-starter.zip ec2…amazon.com:/home/ec2-user

(local location and file name ec2 IP address)

1.Create files in workstation for writing code

**Chef generate repo ‘name file’**

**Cd chef-repo**

**Ls -lrt & ls -la**

Move file .chef file to root directory of repository home location of repository

**Cp -r .chef /opt/prem-repo (copying into root locations)**

On .chef file **cd .chef**

Chefadmin.Pem file : password to connect server.

knife file: location of server.

**Connation to server**

* **Knife ssl fetch**
* **Ls .chef**
* **Knife ssl check**

**Chef Client**

Port number 22 workstation To client

PN 443 client To server

We need to install all servers.

Knife bootstrap IP address –ssh-user ec2-user –sudo –identity-file /home/ec2-user/AA.pem file -N hostname(ec2xxx)

Chef UI interphase

Nodes: where you can see all servers ‘knife node list ’

Policy: all code

Administration: permissions

\* knife node list : view no servers \*

**Cookbook: part 1 24:18**

1.Chef generate cookbook name: http server

2.ls-l

* medtadata.rb: all details of cookbook
* recipes: default rb ‘write code here ’

example: this cookbook is being written to install apache web server, start apache create a user and group copy file ,create directory.

*Package’<name od=f package which you need to install>do*

*Action: install/upgrade/uninstall*

*Action: create file/update file/delete file*

*Action: start/ stop/restart/reload*

*Attributes to that resources*

*Name of the package*

*Name of the file, permission, version,*

Package ‘httpd’ do

Action: install

Timeout ”60”

End

**3.Save and upload to server**

Knife cookbook upload ‘cookbook’ name : Knife cookbook upload httpserver.

Check it on web site :ec2######

Cook book name: httpserver

Details, content, permissions.

1. Policy plan “client requires what policys”

Run List: what steps to do or fallow

**Run list** 🡪 Node 🡪 Server

Creating policy plans;

* Knife node list
* Knife node show name “total list ”
* Knife node run\_list add ‘node name’ recipe name [ ‘httpd ::default]

Knife node run\_list add <node name> “recipe[<cookbook>::defult]”

1. **Idempotency:** if its already app present in chef , checf is not to touch that “no same action” like install
2. **Create user**

Package ‘httpd’ do

Action: install

Timeout 60

End

Group ‘httpadmin’do

Action: create

End

User ‘httpadmin’ do

group ‘httpadmin’

shell’/bin/bash’

action: create

comment “this is a new user “

end

**upload:** knife cookbook upload httpserver

**Run:** chef-client

1. **Service**

Start

Service ‘httpd’ do

Action [start : enable /disable]

End

File ‘/var/www/html/index.html’ dfo

Action : create

Mode “0755”

Owner “root”

Group”root”

Content “<html> <title> apache server </title>

Hello apache server, im configured using chef </h1></body></html>”

End

Directory ‘/opt/apachelogs’do

Action : create

Owner “root”

Group”owner”

End

**upload:** knife cookbook upload httpserver

**Run:** chef-client

1. **Not\_if** (if condition is true then its work. like java already install)

**10.Only\_if (if)**

**Environment:**

Production, Stage, QA, Dev

They are separated using network /subnet and firewall. (vpc)

Production dB: connection details of oracle db.

Cron

Link

Reboot

Script

**11.Food critic:** find out common problems & give you suitable answers

What is diff b/w script and bash?

**JENKINS**

CI

Is a software development practice where developers frequently integrate their work with the projects integration branch and create a build.

Jenkins: its open source integration tool. It’s a center of CI system and is connected to the version control system tool, build tool, repository tool testing and production environmental, test automation tool

A screenshot of a cell phone

Description automatically generated

There are 2 branches in Jenkins port 8080

1.LTS Long term support -every 12 weeks

2. Latest build -every week

We can install by war file or rpm files

Start/Stop

To start Jenkins: brew services start jenkins-lts

To restart the Jenkins : brew services restart jenkins-lts

**Create users + Mange + Assign Roles**

1. Create new users
2. Configure users
3. Create and manage users’ roles. Roles strategy plugin – download restart Jenkins
4. Mange Jenkins -> configure global security -> Authorization -> role base strategy

INTERVIEW

Role: 2 to 3 mints

Build release team to setup Jenkins jobs some maven configuration. Handling customer feedback application. description about application

Interaction with data base team & development

Write code on chef Jenkins -SVM, git check out.

Day to day activities:

checking system status of server-monitor

Memory issues, applications or server down

Daily participate in meetings.

1. Application is java based
2. Written in latest tech angular .net

Setting up a lot of Jenkins jobs on basis of requirements.

Approach, pre-reequipments, CPU, RAM, file systems

Every developer has their own style of writing code with different visions. we prepare for that kind of problems.

Install – trigger Gmail

Backup jobs, nightly jobs hardware requirement

Remedy?

Testing:

Push to production

Project pipeline: continues integration

Java

Maven

Middleware team tomcat/WebLogic

Db Team: setup Jenkins job and triggers permissions chef Jenkins docker (\*\*\*) docker file from run cmd write build & docker stack for deploy.

Docker compose using images, network and volumes

How you secure : ssl certifies .https://

Load balancers: round robin algorithm,

AWS

Need to talk with all teams collaboration (version tools, , , )

Are you have as DevOps: we need customers happy efficiency.

Time/cost.

Number of frequency of software release.

How fast we can find bugs P1 ticket. Outage “bugs”

Tools:

What was your Migration strategy?

Data base

What data base instance you deployed &why?

A: Oracle or Amazon.

Scripting: shell monitor app backup logs, trigger, restart.

More then depended upon requirement I check back with my lead or google to finish that task.

Challenging task & accomplishment on job?

data base migrations

1. 11g to 12c playbook, shell, project documentation, application has without downtime

2. Local to private to AWS.

Important job:

Jenkins pipeline, playbook, tomcat, patching, upgrading, initial application user accounts, compliance.

DR Strategy?

Disaster strategy in cross region. Different place like backup

Continually sync data.

Scripts written for status of production

Every Quarter we winterly bring down some service, if you coming up DR exercise.

Move to cloud:

Benefits : AWS provide latest process Hybrid App server -web server

Why do I need DevOps:

Deployment automation

Testing automation

GIT to Jenkins

[Simply Explained - Savjee](https://www.youtube.com/channel/UCnxrdFPXJMeHru_b4Q_vTPQ)

2.https://youtu.be/pts8zdHel5E

Generate SSH key on git

ON terminal:

ssh-keygen -t rsa -b 4096 -C "buddaprem93@gmail.com"

you will see to keys public / private

eval "$(ssh-agent -s)”

ssh-add ~/.ssh/id\_rsa

go to git up account -> SSH& GPG keys

pbcopy < ~/.ssh/id\_rsa.pub

Compile

Test

Integration

Package