Software Engineering Lab Cheatsheet

Git Configuration

```
git --version
git config --global user.name “NAME”
git config --global user.email “EMAIL”
```

Git Commands

```
git init
git add .
git commit -m "Message"

git help

git log
git log --author="username"
git log --oneline
git status

git config --global --edit
git config --global --list

git diff
git diff --staged
git diff --oneline

git revert <commit_hash>
git reset <commit_hash>

git branch
git branch <name>
git branch -d <name>
git branch -m <old> <new>
git checkout <name>
git checkout -b <name>
git merge <name> (From parent branch)
```

Terminal Commands

```
pwd
mkdir
cd
notepad filename
ls
ls -latr
```

GitHub Commands

```
git remote -v
git remote add origin <URL>
git push origin main

ssh-keygen -t rsa -C <email>
cat ~/.ssh/id_rsa.pub
```

Maven

- New File
- Next
- org.apache.maven.archetypes.quickstart / webapp
- Give Artifact ID
- Dependency in pom.xml (Tomcat Web)
- App.java / index.jsp
- Clean, Install, Test, Build (CIT Goals)
- Run App.java / index.jsp (Tomcat Web)
- Push to GitHub

Jenkins

- Freestyle project
- Provide Git URL
- Build Step - Invoke Top level maven Targets - Clean & Install
- Archive Artifacts **/*
- Post Build Actions - Build Other Projects - Test Project
- Test project - Delete workspace before build starts
- Build Step - Copy artifacts from another project **/* - Latest Successful Build
- Invoke Top Level Maven targets - Test
- Add Post Build Actions - Archive the Artifacts **/*
- Deploy Project - Copy Artifacts - For Web
- Post Build - Deploy war/ear to a container
- Path - webpath
- Select Tomcat version in Container
- Add Credentials
- Build Pipeline View - Up / Down stream config - Initial job
- Run Initial Project
- Go to /manager (Web Project)

Jenkins Script

```
pipeline {
    agent any
    stages {
        stage("Hello") {
            steps {
                echo "Hello"
            }
        }
        stage("Hi") {
            steps {
                echo "Hi"
            }
        }
    }
}
```

Build Script

```
pipeline {
    agent any

    tools{
        maven 'MAVEN_HOME'
        git 'Default'
        jdk 'JAVA_HOME'
    }

    stages {
        stage('git repo & clean') {
            steps {
                bat "if exist mavendemo ( rmdir /s /q mavendemo )"
                bat "git clone https://github.com/username/mavendemo.git"
                bat "mvn clean -f mavendemo"
            }
        }
        stage('install') {
            steps {
                bat "mvn install -f mavendemo"
            }
        }
        stage('test') {
            steps {
                bat "mvn test -f mavendemo"
            }
        }
        stage('package') {
            steps {
                bat "mvn package -f mavendemo"
            }
        }
    }
}
```

Docker

```
docker -v
docker run -d -p 80:80 docker/getting-started
docker run hello-world

docker pull ubuntu
docker run -ti -d ubuntu
docker ps -a
docker exec -it <hash> bash
docker commit <hash> username/ubuntu
docker images
docker login
docker push saikrishna7004/ubuntu
```

Docker File

App.js

```
console.log("Hello World")
```

Dockerfile

```
FROM node:alpine

WORKDIR /name
COPY . /name

CMD node app.js
```

Bash

```
docker build -t myimage:1.0
docker images
```

Docker Compose - Wordpress & MySQL

docker-compose.yml

```
version: '3'

services:
  # Database
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: password
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
    networks:
      - wpsite

  # Wordpress
  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    ports:
      - '8000:80'
    volumes: ['./:/var/www/html']
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
    networks:
      - wpsite
networks:
  wpsite:
volumes:
  db_data:
```

Bash

```
docker-compose up -d
docker-compose stop
```

Kubernetes

```
minikube start --driver=docker
minikube status
kubectl create deployment mynginx --image=nginx
kubectl get deployment
kubectl get pods
kubectl describe pods
kubectl get deployment
kubectl get pods
kubectl scale deployment mynginx --replicas=4
kubectl get deployment
kubectl get pods
kubectl describe pods <pod-name>
kubectl expose deployment mynginx --type=NodePort --port=80
minikube service mynginx --url
minikube dashboard
kubectl delete deployment mynginx
minikube stop
```

Nagios

```
docker pull jasonrivers/nagios:latest
docker run --name nagios4 -p 8888:80 jasonrivers/nagios:latest
```

Username: nagiosadmin
Password: nagios

AWS

Set Up Account

- AWS Free Tier
- Sign Up
- Contact info
- Billing Info
- Confirm Identity
- Basic Support - Free
- Go to AWS Management Console

EC2 Ubuntu

- AWS Management Console
- Services -> All Services -> View All Services
- Navigate to EC2
- Change region to Asia Pacific (Mumbai).
- Resources -> Instances - View running instances.
- Launch new instance.
- Configure instance details - Name, OS (Ubuntu), Instance Type, Network Settings, Storage, etc.
- Create a new key pair for SSH access.

EC2 Web

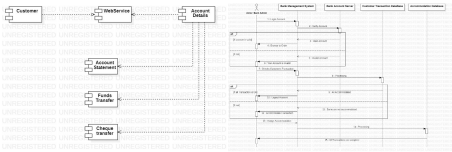
- Launch instance and connect using SSH.
- Update Ubuntu, install Docker, Git, and Nano editor.
- Write index.html
- Initialize Git repo, add files, commit, and push to GitHub.
- Clone GitHub repo to the AWS instance.
- Dockerfile for NGINX web server.
- Build Docker image and run container.
- Access deployed web app using public IP address.
- Stop running Docker container.
- Terminate EC2 instance.
- Delete AWS account if needed.

EC2 Web SSH Bash

```
ssh -i web.pem <URL>
sudo apt update
sudo apt-get install docker.io
sudo apt install git
sudo apt install nano
git clone <URL>
cd <name>
nano Dockerfile
sudo docker build -t mywebapp
sudo docker run -d -p 80:80 mywebapp:latest
sudo docker ps
sudo docker stop <hash>
exit
```

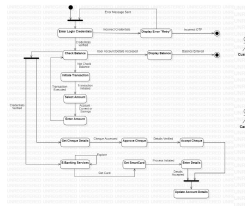
EC2 Web Docker File

```
FROM alpine:nginx
COPY . /usr/share/nginx/html
```

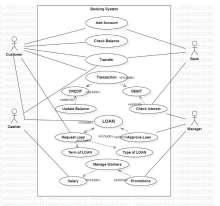


Component

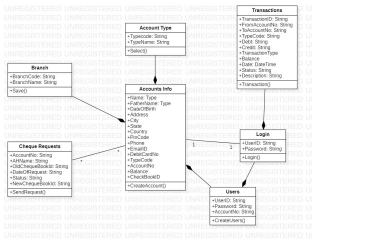
Sequential



State Chart



Use case



Class

Software Engineering Lab Cheatsheet

Git Configuration

```
git --version
git config --global user.name “NAME”
git config --global user.email “EMAIL”
```

Git Commands

```
git init
git add .
git commit -m "Message"

git help

git log
git log --author="username"
git log --oneline
git status

git config --global --edit
git config --global --list

git diff
git diff --staged
git diff --oneline

git revert <commit_hash>
git reset <commit_hash>

git branch
git branch <name>
git branch -d <name>
git branch -m <old> <new>
git checkout <name>
git checkout -b <name>
git merge <name> (From parent branch)
```

Terminal Commands

```
pwd
mkdir
cd
notepad filename
ls
ls -latr
```

GitHub Commands

```
git remote -v
git remote add origin <URL>
git push origin main

ssh-keygen -t rsa -C <email>
cat ~/.ssh/id_rsa.pub
```

Maven

- New File
- Next
- org.apache.maven.archetypes.quickstart / webapp
- Give Artifact ID
- Dependency in pom.xml (Tomcat Web)
- App.java / index.jsp
- Clean, Install, Test, Build (CIT Goals)
- Run App.java / index.jsp (Tomcat Web)
- Push to GitHub

Jenkins

- Freestyle project
- Provide Git URL
- Build Step - Invoke Top level maven Targets - Clean & Install
- Archive Artifacts **/*
- Post Build Actions - Build Other Projects - Test Project
- Test project - Delete workspace before build starts
- Build Step - Copy artifacts from another project **/* - Latest Successful Build
- Invoke Top Level Maven targets - Test
- Add Post Build Actions - Archive the Artifacts **/*
- Deploy Project - Copy Artifacts - For Web
- Post Build - Deploy war/ear to a container
- Path - webpath
- Select Tomcat version in Container
- Add Credentials
- Build Pipeline View - Up / Down stream config - Initial job
- Run Initial Project
- Go to /manager (Web Project)

Jenkins Script

```
pipeline {
    agent any
    stages {
        stage("Hello") {
            steps {
                echo "Hello"
            }
        }
        stage("Hi") {
            steps {
                echo "Hi"
            }
        }
    }
}
```

Build Script

```
pipeline {
    agent any

    tools{
        maven 'MAVEN_HOME'
        git 'Default'
        jdk 'JAVA_HOME'
    }

    stages {
        stage('git repo & clean') {
            steps {
                bat "if exist mavendemo ( rmdir /s /q mavendemo )"
                bat "git clone https://github.com/username/mavendemo.git"
                bat "mvn clean -f mavendemo"
            }
        }
        stage('install') {
            steps {
                bat "mvn install -f mavendemo"
            }
        }
        stage('test') {
            steps {
                bat "mvn test -f mavendemo"
            }
        }
        stage('package') {
            steps {
                bat "mvn package -f mavendemo"
            }
        }
    }
}
```

Docker

```
docker -v
docker run -d -p 80:80 docker/getting-started
docker run hello-world

docker pull ubuntu
docker run -ti -d ubuntu
docker ps -a
docker exec -it <hash> bash
docker commit <hash> username/ubuntu
docker images
docker login
docker push saikrishna7004/ubuntu
```

Docker File

App.js

```
console.log("Hello World")
```

Dockerfile

```
FROM node:alpine

WORKDIR /name
COPY . /name

CMD node app.js
```

Bash

```
docker build -t myimage:1.0
docker images
```

Docker Compose - Wordpress & MySQL

docker-compose.yml

```
version: '3'

services:
  # Database
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: password
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
    networks:
      - wpsite

  # Wordpress
  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    ports:
      - '8000:80'
    volumes: ['./:/var/www/html']
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
    networks:
      - wpsite
networks:
  wpsite:
volumes:
  db_data:
```

Bash

```
docker-compose up -d
docker-compose stop
```

Kubernetes

```
minikube start --driver=docker
minikube status
kubectl create deployment mynginx --image=nginx
kubectl get deployment
kubectl get pods
kubectl describe pods
kubectl get deployment
kubectl get pods
kubectl scale deployment mynginx --replicas=4
kubectl get deployment
kubectl get pods
kubectl describe pods <pod-name>
kubectl expose deployment mynginx --type=NodePort --port=80
minikube service mynginx --url
minikube dashboard
kubectl delete deployment mynginx
minikube stop
```

Nagios

```
docker pull jasonrivers/nagios:latest
docker run --name nagios4 -p 8888:80 jasonrivers/nagios:latest
```

Username: nagiosadmin
Password: nagios

AWS

Set Up Account

- AWS Free Tier
- Sign Up
- Contact info
- Billing Info
- Confirm Identity
- Basic Support - Free
- Go to AWS Management Console

EC2 Ubuntu

- AWS Management Console
- Services -> All Services -> View All Services
- Navigate to EC2
- Change region to Asia Pacific (Mumbai).
- Resources -> Instances - View running instances.
- Launch new instance.
- Configure instance details - Name, OS (Ubuntu), Instance Type, Network Settings, Storage, etc.
- Create a new key pair for SSH access.

EC2 Web

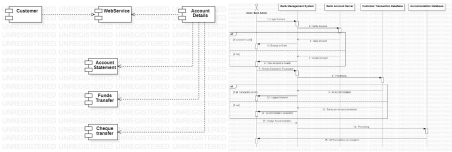
- Launch instance and connect using SSH.
- Update Ubuntu, install Docker, Git, and Nano editor.
- Write index.html
- Initialize Git repo, add files, commit, and push to GitHub.
- Clone GitHub repo to the AWS instance.
- Dockerfile for NGINX web server.
- Build Docker image and run container.
- Access deployed web app using public IP address.
- Stop running Docker container.
- Terminate EC2 instance.
- Delete AWS account if needed.

EC2 Web SSH Bash

```
ssh -i web.pem <URL>
sudo apt update
sudo apt-get install docker.io
sudo apt install git
sudo apt install nano
git clone <URL>
cd <name>
nano Dockerfile
sudo docker build -t mywebapp
sudo docker run -d -p 80:80 mywebapp:latest
sudo docker ps
sudo docker stop <hash>
exit
```

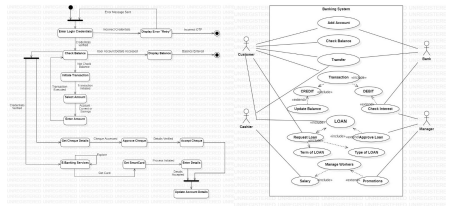
EC2 Web Docker File

```
FROM alpine:nginx
COPY . /usr/share/nginx/html
```



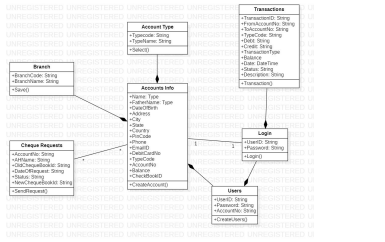
Component

Sequential



State Chart

Use case



Class