**DEVOPS LIFECYCLE**

* Plan
* Develop
* Build
* Test
* Release
* Deploy
* Operate
* Monitor

**PLAN:**

This phase involves planning a roadmap for the success of the project. It also involves the tools required to meet the goal of the project. **Tools:**  **Git, JIRA**



**GIT:**

1. Tracks History, Scalable
2. Open source, Branching is easier
3. Distributed development

**JIRA:**

1. Allows to set up and customize CD pipeline as per individual needs
2. Easy Configuration. Extensible
3. More plugins to integrate

**DEVELOP:**

This phase involves writing the code for the project. It involves detailed software design and creation of software code. **Tools: Stash, GitHub, GitLab**



**STASH:**

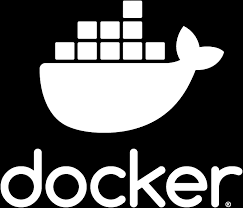
1. It takes current staged and unstaged changes in the branch and saves the work, then returns your branch to the state of latest commit.
2. Used to manage repositories from Git and also other version control system like Mercurial.

**GITHUB And GITLAB:**

1. Source code management, Automation, Secure
2. Helps track and retrieve more information about the commits that have been deployed to an environment.
3. GitLab is a web based Git repository that provides free open and private repositories.

**BUILD:**

In this phase, the code is then build in development environment for testing and bug fixes. **Tools: Docker, Puppet, Chef**



**DOCKER:**

1. Free and open source containerization platform that enables the team to package the code into container that are required to run the code in any environment.
2. Caching a cluster of containers
3. Flexible resource sharing
4. Scalable – many containers can be places in single host
5. Running service on hardware that is much cheaper than standard server.

**PUPPET:**

1. It applies a continuous delivery model to the software release cycle by automating operations and deployment workflow.
2. System administration tasks are saved as manifests and modules.
3. Reusable configuration construction blocks to speedily create preconfigured servers.

**CHEF:**

1. Easily manageable with less manpower.
2. Maintains blueprint of entire infrastructure.
3. Integrates with all major cloud service providers
4. Centralized management – a single Chef server can be used as the centre for deploying policies.

**TEST:**

This phase involves testing of the build code for optimal code quality and bugs identification. **Tools: Vagrant, BlazeMeter, Junit.**

**VAGRANT:**

1. Helps building and managing VM environments in a single workflow.
2. Easy to use workflow and focus on automation.
3. Lowers development environment setup time and increases production parity.

**BLAZEMETER:**

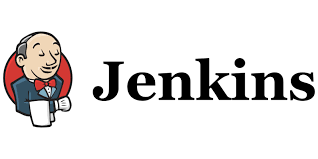
1. Includes complete continuous testing capabilities deeply integrated into single, intuitive workflow.
2. Developers can continue to use their IDE and run tests as code

**JUNIT:**

1. Provides annotation to identify test methods
2. Provides assertion for testing expected results
3. Allows you to write code faster, which increases quality.

**RELEASE:**

This stage involves Ops team to confirm that the software is ready to be released and build into production environment. It is critical since it is the last stop after multiple stages for checks like vulnerabilities and bugs before deployment. **Tools: Jenkins, SonarQube, CircleCI**



**JENKINS:**

1. It will build the code by combining any SCM with any build system.
2. Provides vast number of plugins
3. Provides pipeline support
4. Fast release cycles

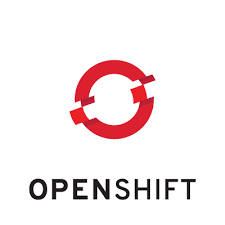
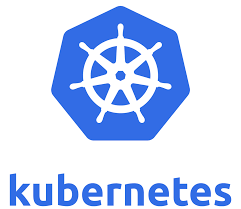
**SONARQUBE:**

1. Code Quality Assurance tool that collects and analyse source code.
2. Combines static and dynamic analysis tools and enables quality to measured continually over time.

**CIRCLECI:**

1. Automate build across multiple environments.
2. Secure pipeline
3. Self hosted runner or private server

**DEPLOY:**

This phase enables the end users to access the build software. This is where approved changes gets deployed to end users/customers. **Tools: Kubernetes, OpenShift, OpenStack**



**KUBERNETES:**

1. Helps build entire infrastructure as code
2. Can automatically provision all aspects of application and tools including access control, networking, databases, storage and security
3. Provides Autoscaling, Lifecycle Management, Loadbalancing

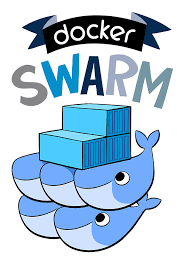
**OPENSHIFT:**

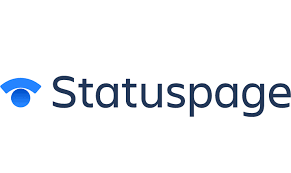
1. Run cloud native microservices at scale.
2. Access a wide ecosystem of partners with open source and cloud native technologies
3. Automated installation and upgrades

**OPENSTACK:**

1. Hybrid Cloud Security, Virtual Network Firewall, DDoS Protection
2. Cloud native protection
3. Web Application/ API protection
4. SaaS security

**OPERATE:**

This is the stage where Ops team will configure and manage project in production environment. **Tools: DockerSwarm, Opsgenie, Statuspage**



**DOCKERSWARM:**

1. Auto scaling, Multi Host networking
2. Service Discovery
3. Load balancing

**OPSGENIE:**

1. Multiple Alerting Channels, Alert Enrichment,
2. Custom Alert Actions, Alert Customization and Classification
3. Alert Lifecycle Tracking

**STATUSPAGE:**

1. Eliminate duplicate support tickets and clunky email lists
2. Level up your incident communication
3. Showcase reliability

**MONITOR:**

This phase involves identifying and collecting information about different issues after software release in production. **Tools: Sensu, Prometheus, Nagios**



**SENSU:**

1. Health checks, custom metrics and logs
2. Comprehensive system health monitoring
3. Alert and Incident management
4. End point management

**PROMETHEUS:**

1. No extensive setup needed
2. Allows fast outage recovery
3. Quick outage discovery

**NAGIOS:**

1. Increased server, services, process and application availability
2. Fast detection of failed server, service, processes and batch jobs
3. Centralized view of entire monitored IT infrastructure.