What is DevOps?

- It's a hot new trend in computing, what’s it?
- A practice of operations and development engineers, participating together through the entire service lifecycle, from the design and development process all the way to production support.
- Model where you have a team that writes the code, another team to test it, yet another team to deploy it, and even another team yet to operate it.
- Also characterized by operations staff, making use of many of the same techniques as developers for their systems work.
- You know, DevOps systems engineering works just like a development workflow.

- All the assets are checked in the source control and have tests associated with them.

- But DevOps, like Agile or Lean is broad enough concept that just a high level definition doesn't really tell you much about what it is.

DevOps is a philosophy that includes
- Collaboration
- Communication
- Sharing
- Openness
- And a holistic approach to software development.

DevOps Principles

1. Incremental Releases
2. Automation
   Infrastructure-as-Code (IaC)
3. DevOps Pipeline
4. Continuous Integration
5. Continuous Delivery
6. Continuous Monitoring
7. Feedback Sharing
8. Version Control
9. Collaboration

DevOps Job Titles

- DevOps or Platform Engineer
- Build Engineer
- Reliability Engineer
- Release Manager
- Data Analyst
- Product Manager

DevOps for present IT

- Cloud Computing
- Version Control System - Git
- What is VCS? and What is SCM?
- Differences of SVN and GIT
- Agile
- Understanding the git stages
- Create branches in Git and Merge & Rebase
- Rename move files and deleting files
- Git repository setup
- Git push, pull and fetch
- Git stash, conflicts, tag

3. CI/CD- Jenkins

- Introduction
- Installation of Jenkins
- System Configuration
- Using Credentials
- SCM poll with
- GitHub/Git

4. Code Testing- SonarQube

- Setting up SonarQube
- Configuring with Git Project
- Running Code scan & Analysis
- Quality Gates
- Adding Testing in CICD
- Pipeline

5. Configuration Management - Ansible

- Overview & Features of Ansible
- Install Ansible
- Ad-hoc commands
- Idempotence
- YAML details
- Writing Playbook
- Ansible flow
- Roles
- Secrets/Ansible Vault
- Network management

6. Terraform (IAC)

- What is Terraform
- What is it used for?
- Difference between Terrform & Ansible
- Terfaform Architecture and commands
- Example configuration file

7. Containerization - Docker

- Containerization Docker concepts
- Docker Architecture
- Docker Image building
- Dockerfile
- Docker Network
- Docker Storage
- Best practices in Docker
- Docker repository

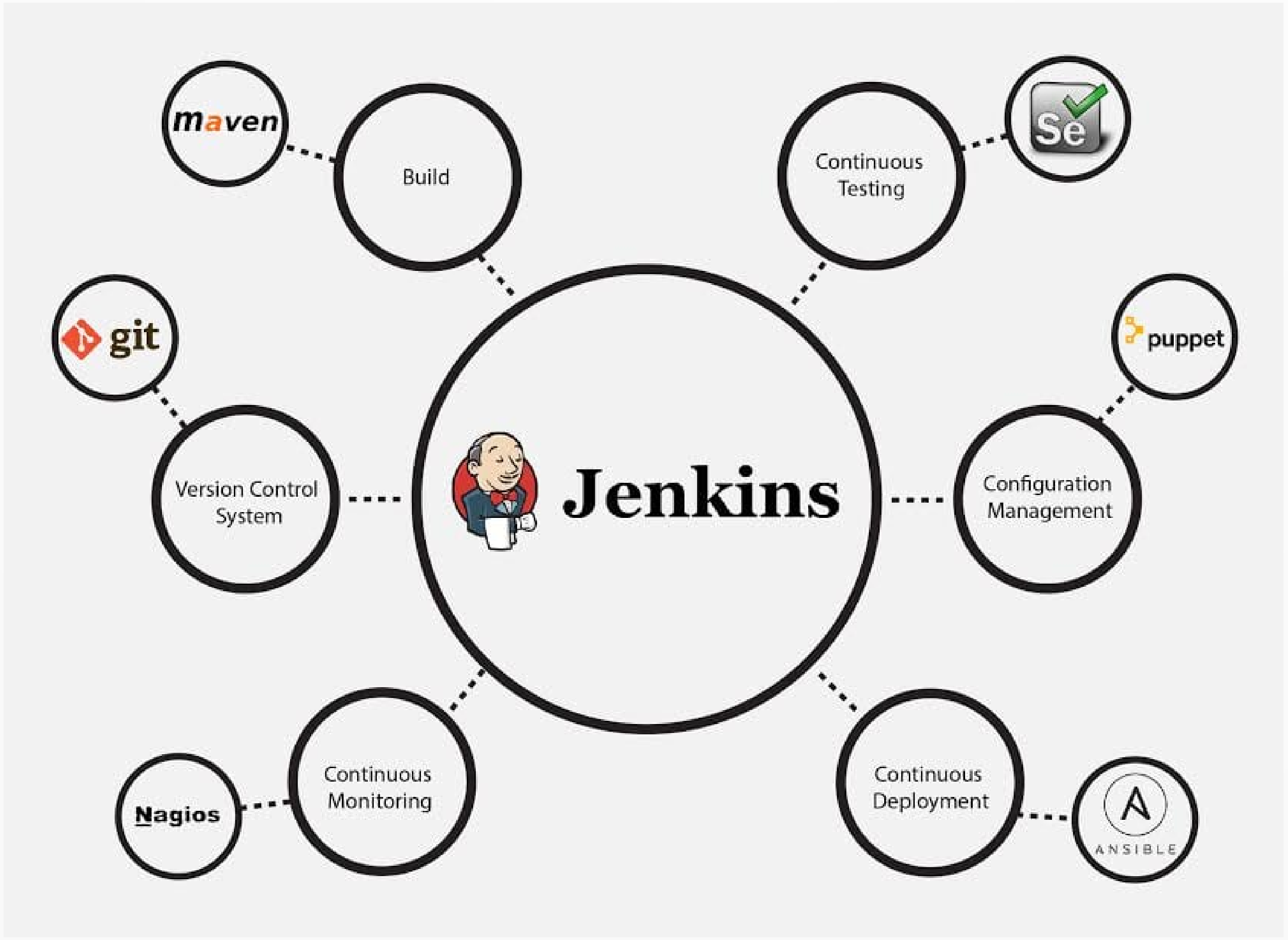
8. Orchestration - Kubernetes

- What is Kubernetes? Why?
- Kubernetes Architecture
- Components of Master, Slave
- Kubernetes Objects
- Kubernetes Installation
- Kubernetes Networking
- Volume in Kubernetes
- Deployment in k8s Replica Sets
- Kubernetes Controllers, Services
- k8s Namespaces, Labels and selectors
- Secretes and ConfigMap
- Setting up Taints, tolerations, Probes
    - RBAC Security model

8. Server & Application Monitoring

- Monitoring Services
- Terminology
- Network Security
- Network Monitoring Tools
- System Requirements Nagios Terminology
- Host Checks & Service Checks
- Event Handlers

- Addons & Plugins Nagios Installation and
- Configuration
- Nagios Installation & Configuration
- Object Configuration
- CGI Configuration


maven
Build
Continuous Testing
Se
git
puppet
Version Control System
Jenkins
Configuration Management
Nagios
Continuous Monitoring
Continuous Deployment
ANSIBLE

What is a Version Control System?

- A system that records all the modifications made to a file or set of data so that a specific version may be called up later if needed.
- The system makes sure that all the team members are working on the file's latest version, and everyone can work simultaneously on the same project.

What is Git?

- Version control system used for tracking changes in computer files, making it a top-rated utility for programmers world-wide.

- Can handle projects of any size.

- Coordinate the work among project team members and track their progress over time.

- It also benefits both programmers and non-technical users by keeping track of their project files.

- Allows multiple users to work together without disrupting each other's work.

- Now that you've been introduced to Git, you have the foundation needed to understand what is GitHub better.

What is GitHub?

- A Git repository hosting service that provides a web-based graphical interface.
- It is the world's largest coding community.
- Putting a code or a project into GitHub brings it increased, widespread exposure.
- Programmers can find source codes in many different languages and use the command-line interface, Git, to make and keep track of any changes.
- Helps every team member work together on a project from any location while facilitating collaboration.
- We can also review previous versions created at an earlier point in time.

What are GitHub's Features

1. Easy Project Management
GitHub is a place where project managers and developers come together to coordinate, track, and update their work so that projects are transparent and stay on schedule.

2. Increased Safety With Packages
Packages can be published privately, within the team, or publicly to the open-source community. The packages can be used or reused by downloading them from GitHub.

3. Effective Team Management
GitHub helps all the team members stay on the same page and organized. Moderation tools like Issue and Pull Request Locking help the team to focus on the code.

4. Improved Code Writing

Pull requests help the organizations to review, develop, and propose new code.

Team members can discuss any implementations and proposals through these before changing the source code.

5. Increased Code Safety

GitHub uses dedicated tools to identify and analyze vulnerabilities to the code that other tools tend to miss.

Development teams everywhere work together to secure the software supply chain, from start to finish.

6. Easy Code Hosting

All the code and documentation is in one place.

There are millions of repositories on GitHub, and each repository has its own tools to help you host and release code.

What is Jenkins?

- Jenkins is an open source continuous integration tool written in Java.
- Provides continuous integration services for software development.
- A server-based system running in a servlet container such as Apache Tomcat.
- Jenkins is an award-winning application that monitors executions of repeated jobs, such as building a Software project or jobs run by cron.

And what is continuous integration?

- A development practice that requires developers to integrate code into a shared repository several times a day.
- Each check-in is then verified by an automated build, allowing teams to detect problems early.

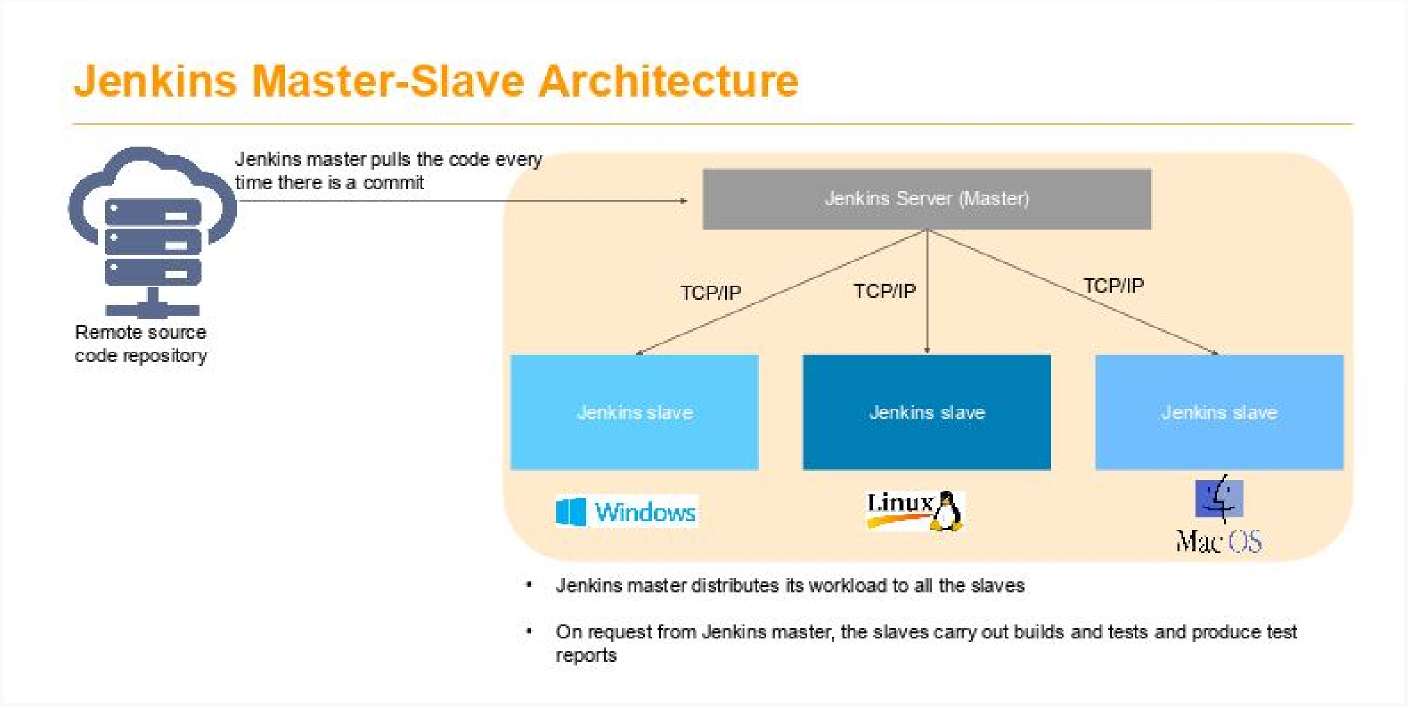
Advantages of Jenkins include:

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

Build Process before Jenkins	Build Process after Jenkins
• Code was committed all at once when all Developers had completed their assigned coding tasks. • deploy the source into the test server and • Later, Build is tested and deployed. • Code commit built, and test cycle was very infrequent, • notifies the deployment team. • and a single build was done after many days	• The code is built and tested as soon as Developer commits code. • Jenkin will build and test code many times during a day And If the build is successful, then the code is deployed into the test server and notifies the deployment team. • If the build fails, then Jenkins will notify the errors to the developer team.
Since the code was built all at once, some developers would need to wait until other developers finish coding to check their build.	The code is built immediately after any of the Developer commits.
Difficult to isolate, detect, and fix errors for multiple commits.	As the code is built after each commit of a single developer, it's easy to detect who's code caused the build to fail.
Code build and test process are entirely manual, so more chances of failure	Automated build and test process saving timing and reduced defects
The code is deployed once all the errors are fixed and tested.	The code is deployed after every successful build and test
Development Cycle is slow	The development cycle is fast. New features are more readily available to users. Increases profits

What are the Jenkins Features?

- Easy Installation/ Configuration
- Available Plugins
- Extensible
- Easy Distribution
- Free Open Source


Jenkins Master-Slave Architecture
Jenkins master pulls the code every time there is a commit
Remote source code repository
Jenkins Server (Master)
TCP/IP
TCP/IP
TCP/IP
Jenkins slave
Jenkins slave
Jenkins slave
Windows
Linux
Mac OS

- Jenkins master distributes its workload to all the slaves
- On request from Jenkins master, the slaves carry out builds and tests and produce test reports

Before you start - Jenkins and SCM

- Jenkins and configuration management tools like Chef and Puppet go hand in hand.
- The reason for it is to have consistent environments.
- You should script out your Jenkins install and configuration. Also useful when using nodes.

What is Maven?

- The Maven project is developed by Apache Software Foundation where it was formerly a part of the Jakarta project.
- A powerful build automation tool that is primarily used for Java-based projects that helps tackle two critical aspects of building software –
    1. Describes how software is built
    2. Describes the dependencies.
- Dynamically downloads Java libraries and Maven plug-ins from one or more repositories such as the Maven Central Repository and stores them in a local cache.
- The artifacts of the local projects can also be updated with this local cache.
- Help us build and manage projects written in C#, Ruby, Scala, and other languages.

- Project Object Model(POM) file is an XML file that contains information related to the project and configuration information such as dependencies, source directory, plugin, goals, etc. used by Maven to build the project.
- When you execute a maven command you give maven a POM file to execute the commands.
- Maven reads the pom.xml file to accomplish its configuration and operations.

When to use Maven?

- If there are too many dependencies for the project.
- When the dependency versions update frequently.
- Continuous builds, integration, and testing can be easily handled by using maven.
- When one needs an easy way to generate documentation from the source code, compiling the source code, packaging compiled code into JAR files or ZIP files

Maven life cycle, phases and goals

Maven life cycle

- Maven follows a life cycle deploy and distribute the target project.
- There are three built-in life cycles:
- default – This is the main life cycle of Maven as it is responsible for project deployment.
- clean – This life cycle is used to clean the project and remove all files generated by the previous build.
- site – The aim of this life cycle is to create the project's site documentation.

Each life cycle is made up of a sequence of phases.

- The default build life cycle consists of 23 phases as it is the main build life cycle
- Maven clean life cycle consists of 3 phases
- Site life cycle is made up of 4 phases.

What Is Code Quality? And How to Improve Code Quality

Code quality defines code that is good (high quality) — and code that is bad (low quality).

Essentially, code that is considered good:

- Does what it should.
- Follows a consistent style.
- It is easy to understand.
- Has been well-documented.
- It can be tested.
- Checked for potential bugs and performance, security, or vulnerabilities issues
- Is the code duplicated anywhere
- Logical sense, or is it too complex
- Does the code have unit tests
- Does the code follow good software design and architecture principles.

Static Code Analysis:

- Static code analysis is done without executing any of the code.
- It is a collection of algorithms and techniques to analyze source code to automatically find potential errors and poor coding practices.
- This is done with compiler errors and run-time debugging techniques such as white box testing. Static code analysis is also considered a way to automate code review process.
- The tasks involved in static code analysis can be divided as such

What is SonarQube?

- SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality.
- Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications.
- It supports 25+ major programming languages through built-in rulesets and can also be extended with various plugins.

Benefits of SonarQube

Sustainability - Reduces complexity, possible vulnerabilities, and code duplications, optimising the

life of applications.

Increase productivity - Reduces the scale, cost of maintenance, and risk of the application; as such,

it removes the need to spend more time changing the code

Quality code - Code quality control is an inseparable part of the process of software development.

- **Detect Errors –** In the code and alerts developers to fix them automatically before submitting them for output.
- **Increase consistency -** Determines where the code criteria are breached and enhances the quality
- **Business scaling -** No restriction on the number of projects to be evaluated
- **Enhance developer skills** - Regular feedback on quality problems helps developers to improve their coding skills
- **Quality Gates:** Is the best way to ensure that standards are met and regulated across all the projects in our organization.
- Quality Gates can be defined as a set of threshold measures set on your project.

- Few conditions that can be in included are listed below.
- Code Coverage > certain value
- Number of Blocker issues >certain value
- Security Rating / Unit Test Pass Rate etc..

Why use configuration management tools?

- Configuration management tools enable changes and deployments to be faster, repeatable, scalable, predictable, and able to maintain the desired state, which brings controlled assets into an expected state.
- Some advantages of using configuration management tools include:
- Adherence to coding conventions that make it easier to navigate code
- Idempotency, which means that the end state remains the same, no matter how many times the code is executed
- Distribution design to improve managing large numbers of remote servers

What Is Ansible?

Change Management

Provisioning

Automation

Orchestration

Change Management

Define a "System State"

Enforce the System State

System State

Apache Web Installed

Apache Web at version x.xx.x

Apache Web Started

Provisioning

Prepare a system to make it ready

- Transition from one state to a different state

Examples

- Make an FTP Server
- Make an Email Server
- Make a DB Server

Automation

Define tasks to be executed automatically

- Ordered Tasks
- Make decisions
- Ad-hoc tasks

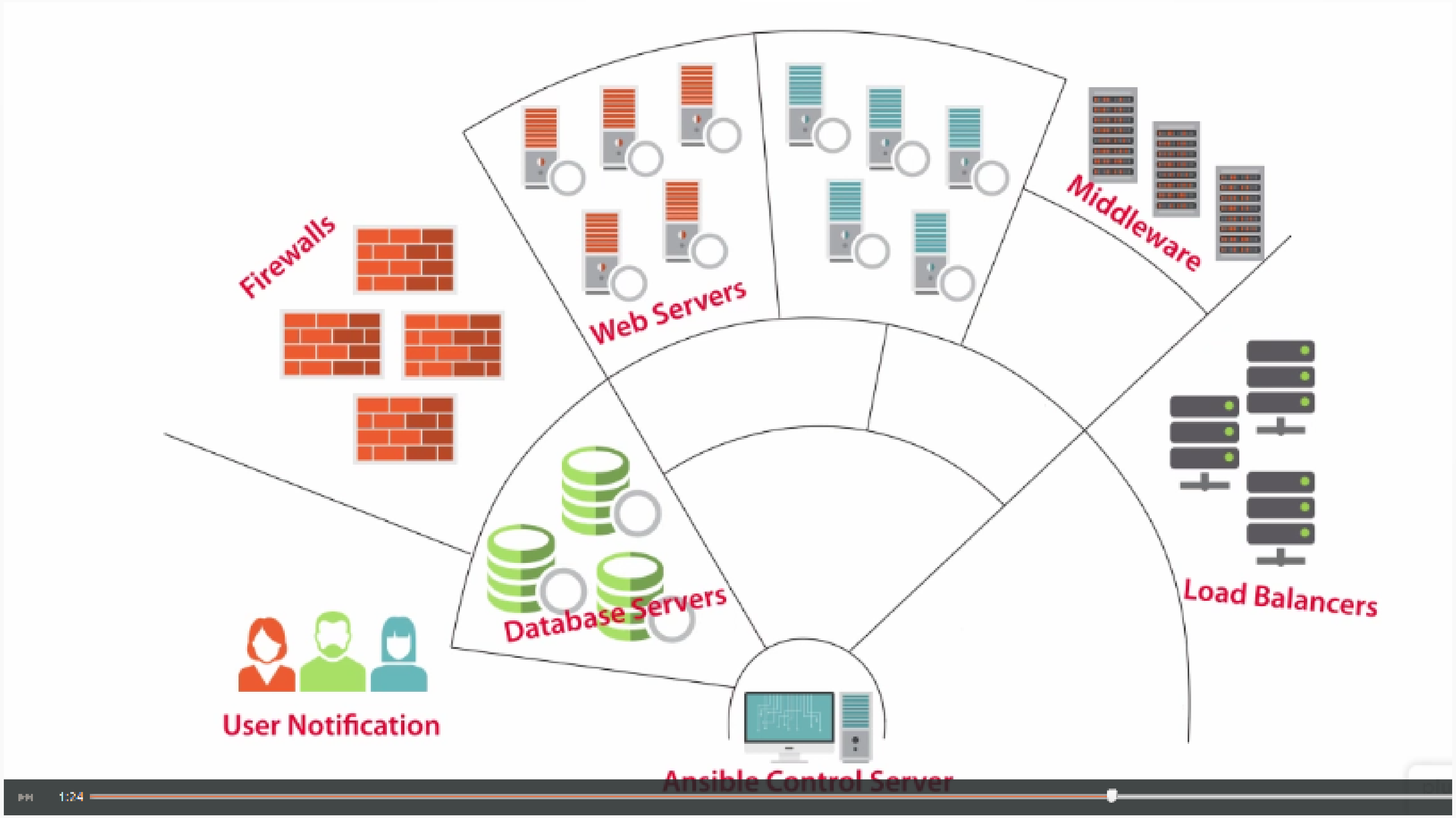
Set it and Forget it

- Run the task
- Get a cup of coffee
- Walk back to desk seeing tasks finished
- Sip your coffee and feel productive

Orchestration

Coordinates automation BETWEEN systems

- Task 1 - System 1
- Task 2 - System 2
- Task 3 - System 3
- Task 4 - System 1


Firewalls
Web Servers
Middleware
Database Servers
Load Balancers
User Notification
Ansible Control Server
1:24

Why Ansible?

What makes it so different?

It's clean!

No agents

No database

No residual software

No complex upgrades

Ansible Execution

YAML

- No programming required
- NOT a markup language
- Structured
- Easy to read and write

Built-in security

- Uses SSH
- Root / Sudo usage
- Encrypted vault
- No PKI needed

Easy to extend

- URL / RESTful calls
- Shell Commands
- Scripts
- Ansible-Galaxy

Inventory

The Inventory is a description of the nodes that can be accessed by Ansible. By default, the Inventory is described by a configuration file, whose default location is in./etc/ansible/hosts The configuration file lists either the IP address or hostname of each node that is accessible by Ansible.

Every host is assigned to a group such as web servers, db servers etc. The inventory file can be in one of many formats such as yaml, INI etc

Example of an Inventory file

```
mail.example.com

[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com
```

Playbook

Playbooks are simple YAML files. These files are descriptions of the desired state of your systems. Ansible then does the hard work of getting your systems to that state no matter what state they are currently in. Playbooks make your installations, upgrades and day-to-day management repeatable and reliable.

Playbooks are simple to write and maintain. Playbooks are written in a natural language so they are very easy to evolve and edit.

Playbook contains Plays.

Plays contain tasks.

tasks call modules.

Example of an ansible playbook

```
---
- hosts: webservers
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum: name=httpd state=latest
  - name: ensure apache is running
    service: name=httpd state=started enabled=yes
```

Modules

There are over 1000 modules provided by Ansible to automate every part of the environment. Modules are like plugins that do the actual work in Ansible, they are what gets executed in each playbook task.

Each module is mostly standalone and can be written in a standard scripting language (such as Python, Perl, Ruby, Bash, etc.). One of the guiding properties of modules is idempotency, which means that even if an operation is repeated multiple times, it will always place the system into the same state.

Example of Modules

There are lots of modules such as :

Service, file, copy, iptables etc.

Any Module can be used as :

```
ansible 127.0.0.1 -m service -a "name=httpd state=started"
ansible localhost -m ping
```

Roles

Roles are a way to group tasks together into one container. We could have a role for setting up MySQL, another one for configuring iptables etc.

Roles makes it easy to configure hosts. Any role can be performed on any host or group of hosts such as:

```
- hosts: all
  roles:
    - role_1
    - role_2
```

What is Terraform

Infrastructure as code (IaC) tool that allows us to

- Build,
- change
- Version infrastructure safely and efficiently.

Low-level components such as

- compute instances
- Storage,
- Networking

High-level components such as

- DNS entries,
- SaaS features, etc.

What is Terraform used for?

- External resource management -- Supports public/private cloud infrastructure, as well as network appliances and SaaS deployments.
- Multi-cloud deployment --Supports multiple cloud services helps increase fault tolerance.
- Multi-tier applications -- Allows each resource collection to easily be scaled up or down as needed.
- Self-service clusters -- The registries make it easy for users to find prepackaged configurations that can be used as is or modified to meet a particular need.
- Software defined networking (SDN) -- Terraform's readability makes it easy for network engineers to codify the configuration for an SDN.
- Resource scheduler -- Terraform modules can stop and start resources on AWS and allow Kubernetes to schedule Docker containers.
- Disposable environments -- modules can be used to create an ad hoc, throwaway test environment for code before it's put into production.

Features of Terraform

Terraform works by building a graph database that provides operators with insight into resource dependencies.

Generates an execution plan that allows operators to see what sequence of steps Terraform will take when a setting is applied or a change is made. Terraform software also includes the following:

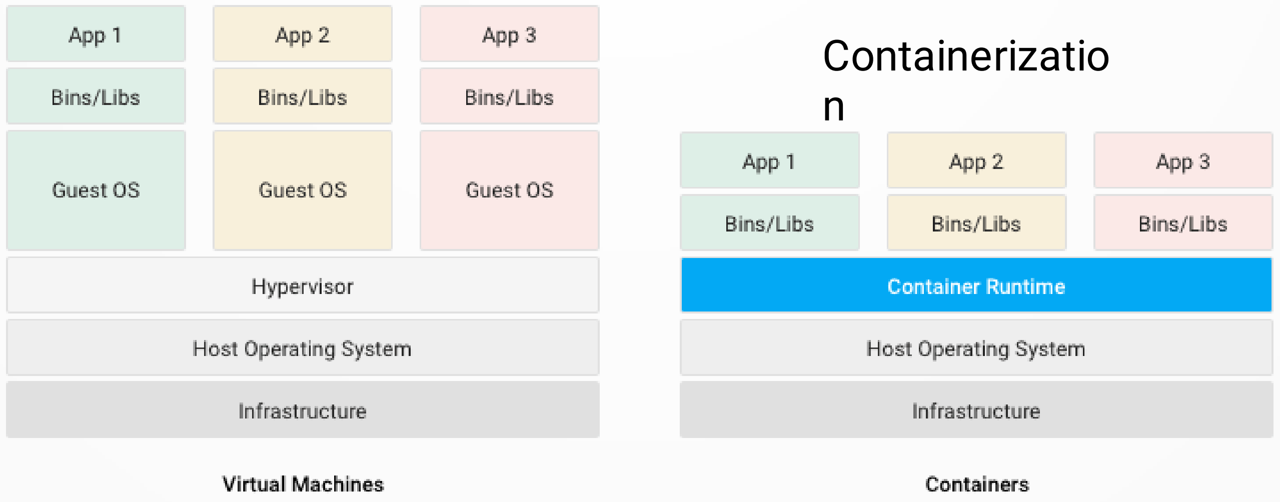
- Console that allows users to observe functions such as numeric, string, date and time, collections as well as encoding functions;
- Configuration language that supports interpolation and enables admins to pass functions as a string to perform a range of operations;
- Offers an ability to translate HCL code into the JSON format; and
- A feature called Module Count that specifies the number of modules that have been applied to an infrastructure.

What are containers?

- Containers are a lighter-weight, more agile way of handling virtualization — since they don't use a hypervisor, you can enjoy faster resource provisioning and speedier availability of new applications.
- Containerization packages together everything needed to run a single application or microservice (along with runtime libraries they need to run).
- The container includes all the code, its dependencies and even the operating system itself.
- This enables applications to run almost anywhere — a desktop computer, a traditional IT infrastructure or the cloud.

Virtualization

Containerizatio
n



Why Containers?

- Build,Ship, and Run any App, Anywhere.
- No more dependency problems;e.x.no RPM/DEB o r dynamic libraries incompatibility issues.
- No more:"but it works on my machine", as the same image used in server deployed to prod.
- Serverless implementations are mainly backed by containers.

Docker Engine

It is the core element of the whole Docker system.
Engine is installed on the host machine and it uses client-server architecture.

Components in the Docker Engine:

Server: It is the docker daemon called dockerd. It can create and manage docker images. Containers, networks, etc.

Rest API: It is used to instruct docker daemon what to do.

Command Line Interface (CLI): It is a client which is used to enter docker commands.

Docker Client:
Docker users can interact with Docker through a client.
When any docker commands runs, the client sends them to dockerd daemon, which carries them out.
Docker API is used by Docker commands.
Docker client can communicate with more than one daemon

Docker - Compose

- Docker Compose is used to run multiple containers as a single service.
- For example, An application with NGINX and MySQL, we can create one file which would start both the containers as a service without the need to start each one separately.
- Benefits of Docker Compose
- Single host deployment - This means you can run everything on a single piece of hardware
- Quick and easy configuration - Due to YAML scripts
- High productivity - Docker Compose reduces the time it takes to perform tasks
- Security - All the containers are isolated from each other, reducing the threat landscape

Basic Commands in Docker Compose

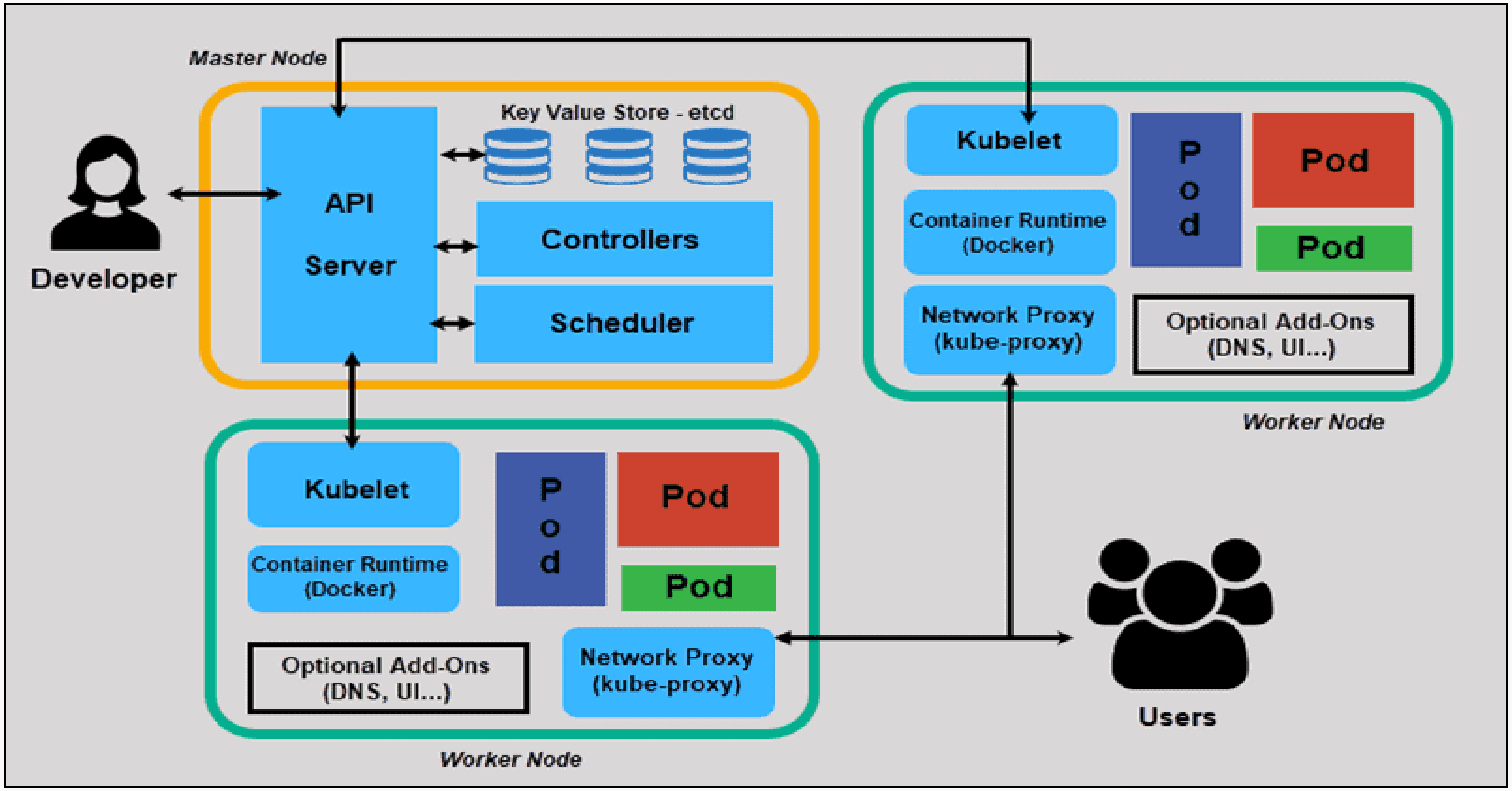
- Start all services: Docker Compose up
- Stop all services: Docker Compose down
- Install Docker Compose using pip: pip install -U Docker-compose
- Check the version of Docker Compose: Docker-compose-v
- Run Docker Compose file: Docker-compose up -d
- List the entire process: Docker ps
- Scale a service - Docker Compose up -d -scale
- Use YAML files to configure application services - Docker Compose.yml

Why Kubernetes?

- To manage hundreds or thousands of containers,on a fleet of hundred or thousands of nodes.
- Deploy an application without worrying about the hardware infrastructure.
- To ensure that applications will stay running according with the specifications.
- Facilitate scaling and upgrades not only for the workloads but for the k8s cluster itself.

WHAT IS KUBERNETES?

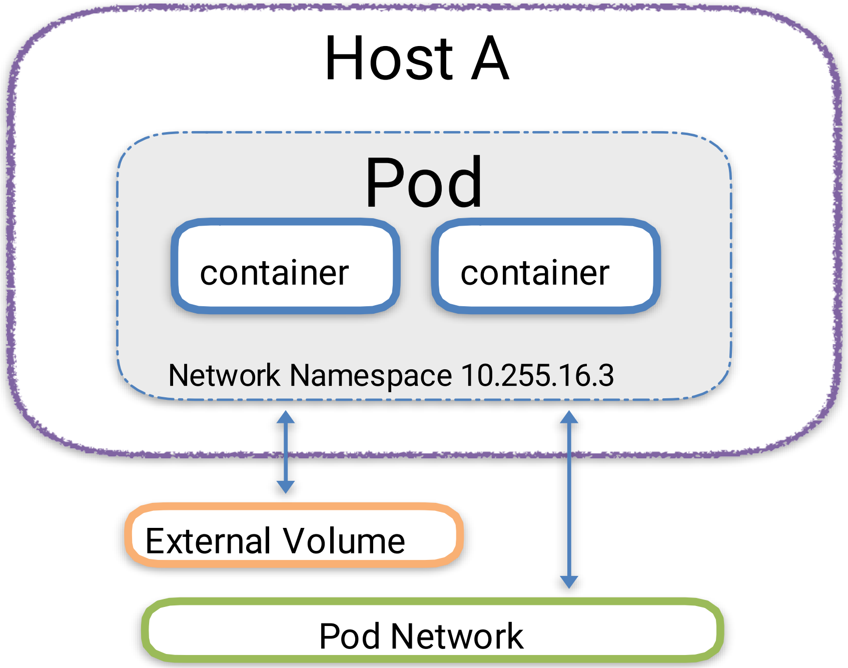
- A distributed cluster technology that manages container- based systems in a declarative manner using an API(a container orchestrator).
- Designed from the ground-up as a loosely coupled collection of components centered around deploying, maintaining and scaling workloads.
- Abstracts away the underlying hardware of the nodes and provides a uniform interface for workloads to be both deployed and consume the shared pool of resources.


Master Node
Key Value Store - etcd
Developer
API Server
Controllers
Scheduler
Kubelet
Container Runtime (Docker)
Network Proxy (kube-proxy)
Pod
Pod
Pod
Optional Add-Ons (DNS, UI...)
Worker Node
Kubelet
Container Runtime (Docker)
Pod
Pod
Pod
Optional Add-Ons (DNS, UI...)
Network Proxy (kube-proxy)
Worker Node
Users

KEY CONCEPTS

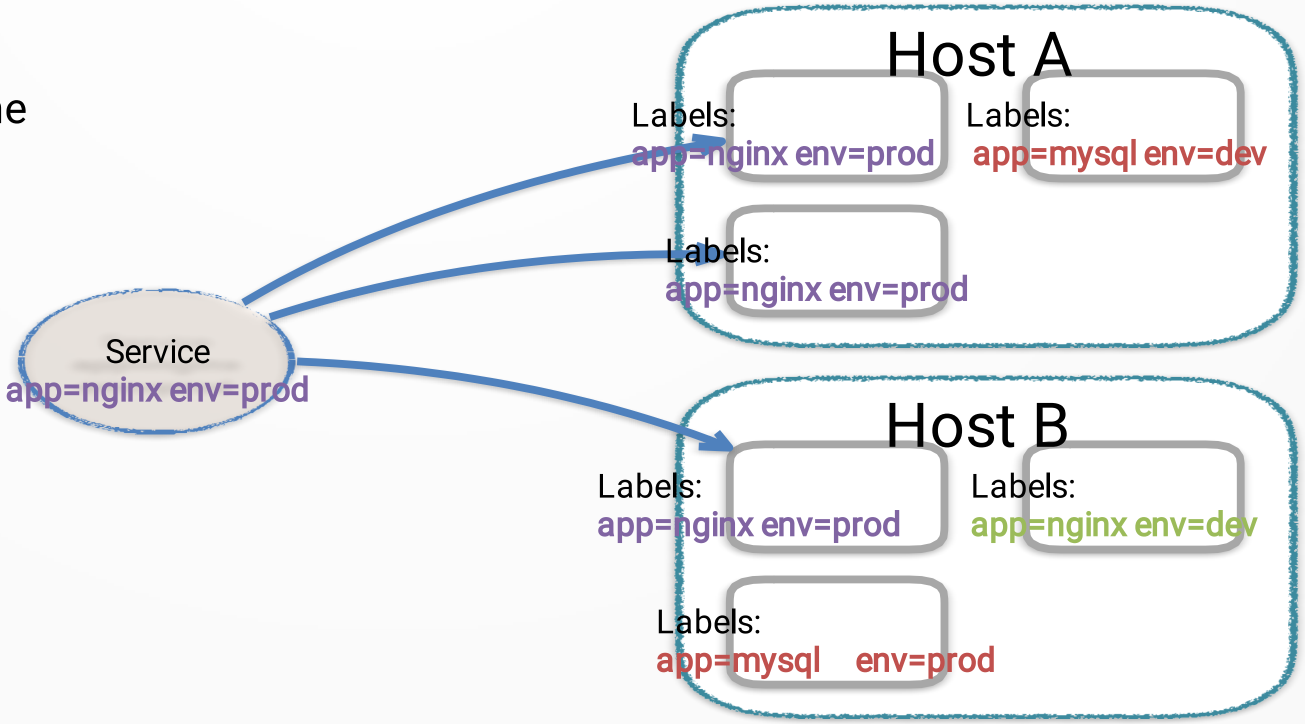
PODS

- Atomic unit or smallest "unit of work" of Kubernetes.
- Pods are one or MORE containers that share volumes,
- a network namespace, and are apart of a single context.

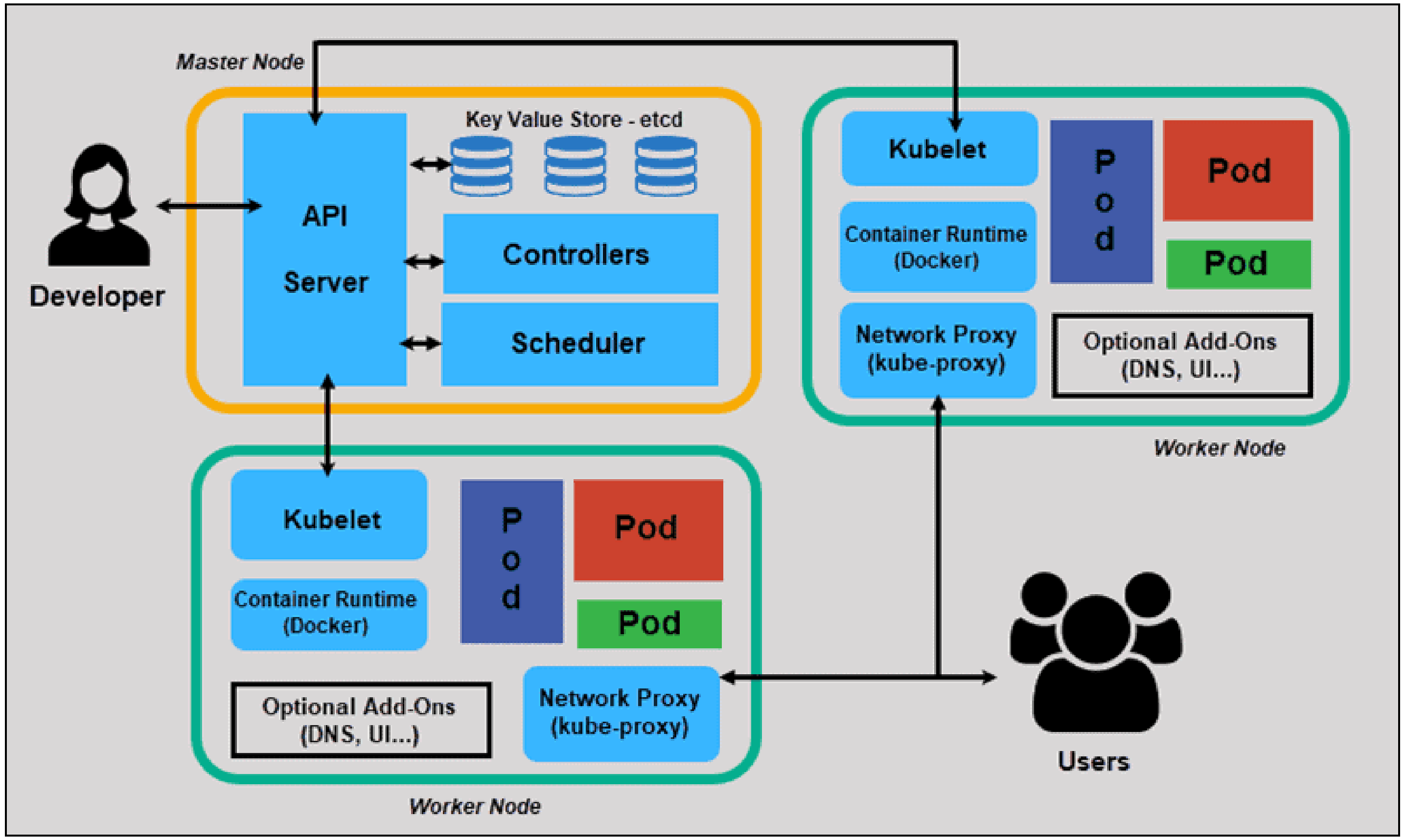


Services

- ✓ Unified method of accessing the exposed workloads of Pods.
- ✓ Durable resource
  - ✓ static cluster IP
  - ✓ static name spaced DNS name



ARCHITECTURE OVERVIEW


Master Node
Key Value Store - etcd
Developer
API Server
Controllers
Scheduler
Kubelet
Container Runtime (Docker)
Network Proxy (kube-proxy)
Pod
Pod
Pod
Optional Add-Ons (DNS, UI...)
Worker Node
Kubelet
Container Runtime (Docker)
Pod
Pod
Pod
Optional Add-Ons (DNS, UI...)
Network Proxy (kube-proxy)
Worker Node
Users

KUBE-APISERVER

- Provides a forward facing REST interface into the kubernetes control plane and datastore.
- All clients and other applications interact with kubernetes strictly through the APIServer.
- Acts as the gatekeeper to the cluster by handling authentication and authorization,request
- validation,mutation,and admission control in addition to being the front-end to the backing datastore.

etcd

- ✓ etcd acts as the cluster datastore.
- ✓ Purpose in relation to Kubernetes is to provide a strong,consistent and highly available key-values to refer
- ✓ persisting cluste rstate.
- ✓ Stores objects and config information.
- ✓ Uses “Raft Consensus” among a quorum of systems to create a fault-tolerant consistent “view” of the cluster.

KUBE-CONTROLLER-MANAGER

- Serves as the primary daemon that manages all core component control loops.

- Monitors the cluster state via the apiserver and steers the cluster towards the desired state.

KUBE-SCHEDULER

- Component on the master that watches newly created pods that have no node assigned, and selects a node for them to run on.
- Factors taken in to account for scheduling decisions include individual and collective resource requirements, hardware/software/policyconstraints, affinity and anti-affinity specifications, data locality, inter-workload interference and deadlines.

- **CLOUD-CONTROLLER-MANAGER**

  Daemon that provides cloud-provider specific knowledge and integration capability into the core control
  loop of Kubernetes.
- The controllers include Node, Route, Service, and add an additional controller to handle things such as
  Persistent Volume Labels.

KUBE-PROXY

- Manages the network rules on each node.
- Performs connection forwarding or loadbalancing for Kubernetes cluster services.

KUBELET

- An agent that runs on each node in the cluster.
- It makes sure that containers are running in a pod.
- The kubelet takes a set of Pod Specs that are provided through various mechanisms and ensures that the containers described in those Pod Specs are running and healthy.

CONTAINER RUNTIME ENGINE

- A container runtime is a CRI (Container Runtime Interface) compatible application that executes and manages containers.
- containerd(Docker)
- cri-o
- rkt
- kata(formerly clear and hyper)
- virtlet(VMCRI compatible runtime)

KUBERNETES NETWORKING

- PodNetwork
- Cluster-widenetwork used for pod-to-pod communication managed by a
- CNI(ContainerNetworkInterface)plugin.
- ServiceNetwork
- Cluster-widerangeofVirtualIPsmanage dby kube-proxy for service discovery.

FUNDAMENTAL NETWORKING RULES

- All containers with in a pod can communicate with each other unimpeded.
- All Pods can communicate with all other Pods without NAT.
- AllnodescancommunicatewithallPods(andvice-versa)withoutNAT.
- TheIPthataPodseesitselfasisthesameIPthatothersseeitas.

https://www.janbasktraining.com/blog/devops-tutorial/
https://github.com/jenkins-docs/simple-java-maven-app/blob/master/jenkins/Jenkinsfile

What is Continuous Monitoring?

- A process to detect, report, respond all the attacks which occur in its infrastructure.
- Once the application is deployed into the server, the role of continuous monitoring comes in to play.
- The entire process is all about taking care of the company's infrastructure and respond appropriately.

What is Nagios?

- An open source software for continuous monitoring of systems, networks, and infra
- It runs plugins stored on a server which is connected with a host or another server on your network or the Internet.
- In case of any failure, Nagios alerts about the issues so that the technical team can perform recovery process immediately.
- Used for continuous monitoring of systems, applications, service and business process in a DevOps culture.

Why We Need Nagios tool?

Here are the important reasons to use Nagios monitoring tool:

- Detects all types of network or server issues
- Helps you to find the root cause of the problem which allows you to get the permanent solution to the problem
- Active monitoring of your entire infrastructure and business processes
- Allows you to monitor and troubleshoot server performance issues
- Helps you to plan for infrastructure upgrades before outdated systems create failures
- You can maintain the security and availability of the service
- Automatically fix problems in a panic situation

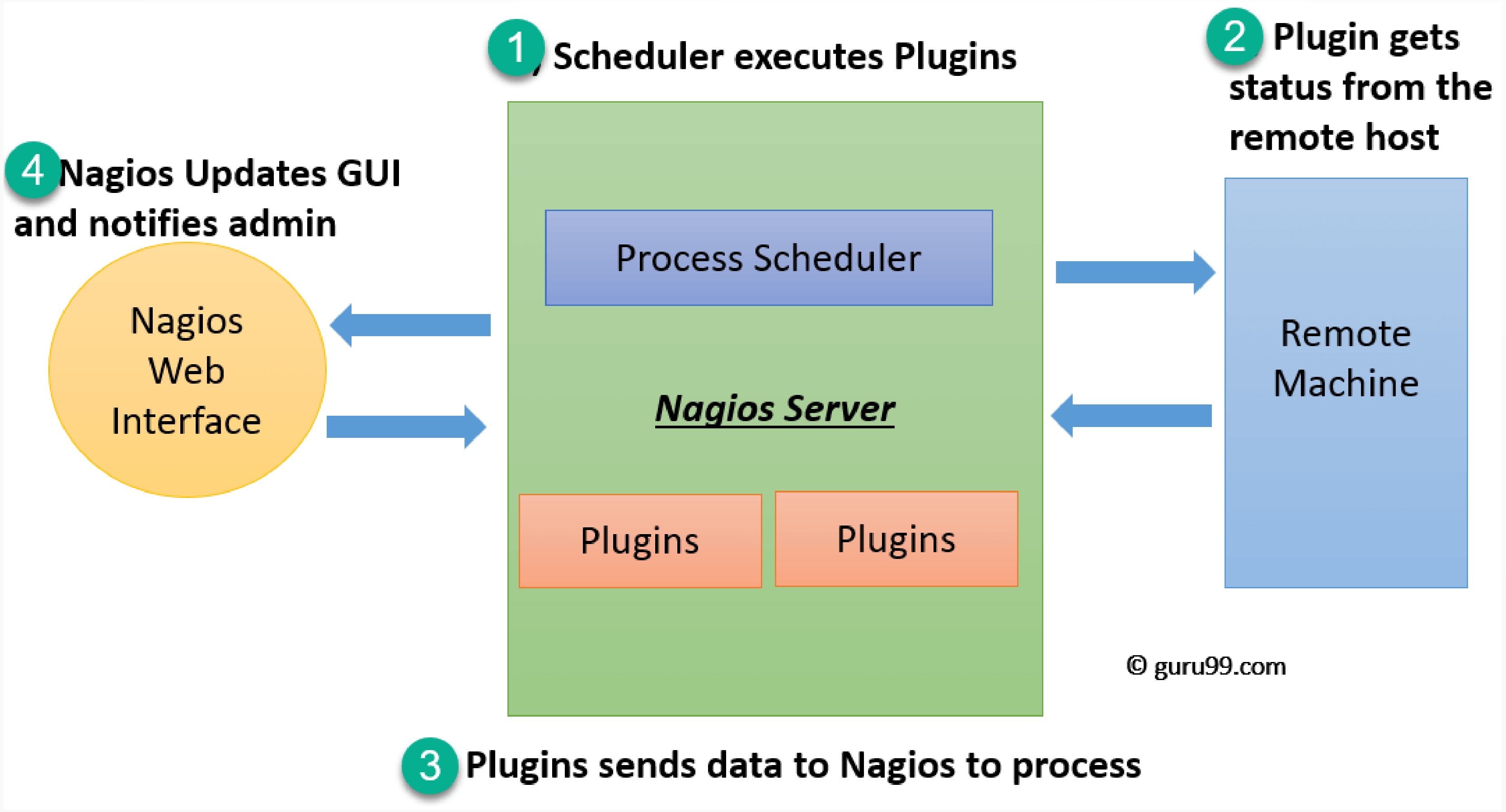
Features of Nagios

Following are the important features of Nagios monitoring tool:

- Relatively scalable, Manageable, and Secure
- Good log and database system
- Informative and attractive web interfaces
- Automatically send alerts if condition changes
- If the services are running fine, then there is no need to do check that host is an alive
- Helps you to detect network errors or server crashes
- You can troubleshoot the performance issues of the server.
- The issues, if any, can be fixed automatically as they are identified during the monitoring process
- You can monitor the entire business process and IT infrastructure with a single pass
- The product's architecture is easy writing new plugins in the language of your choice
- Nagios allows you to read its configuration from an entire directory which helps you to decide how to define individual files
- Monitor network services like HTTP, SMTP, HTTP, SNMP, FTP, SSH, POP, etc.
- Helps you to define network host hierarchy using parent hosts
- Ability to define event handlers which runs during service or host events for proactive problem resolution
- Support for implementing redundant monitoring hosts

Nagios Architecture

Nagios is a client-server architecture. Usually, on a network, a Nagios server is running on a host, and plugins are running on all the remote hosts which should be monitored.


1 Scheduler executes Plugins
2 Plugin gets status from the remote host
3 Plugins sends data to Nagios to process
4 Nagios Updates GUI and notifies admin
Nagios Web Interface
Process Scheduler
Nagios Server
Plugins
Plugins
Remote Machine
© guru99.com

- The scheduler is a component of server part of Nagios. It sends a signal to execute the plugins at the remote host.
- The plugin gets the status from the remote host
- The plugin sends the data to the process scheduler
- The process scheduler updates the GUI and notifications are sent to admins

Plugins

- Nagios plugins provide low-level intelligence on how to monitor anything and everything with Nagios Core.
- Plugins operate acts as a standalone application, but they are designed to be executed by Nagios Core.
- It connects to Apache that is controlled by CGI to display the result. Moreover, a database connected to Nagios to keep a log file.