Viva Questions and Answers – Setup Eclipse for DevOps

- Q: What is Eclipse IDE and why is it used in DevOps?

A: Eclipse is an Integrated Development Environment (IDE) used for developing Java and other applications. In DevOps, it's often used for development, version control, automation scripting, and integrating with tools like Jenkins, Git, Maven, and Docker.

- Q: How do you install Eclipse for DevOps purposes?

A: 1. Download Eclipse IDE for Enterprise Java Developers from the official site.
2. Install Java JDK before installing Eclipse.
3. Launch Eclipse and install additional plugins based on DevOps tools like Git, Maven, Docker, etc.

- Q: Which plugins are useful in Eclipse for DevOps integration?

A: - EGit: Git integration
- Maven Integration (M2E): For build automation
- Jenkins Plugin: Continuous Integration
- Docker Tooling: Manage Docker containers
- SonarLint: Code quality checking

- Q: How do you configure Git in Eclipse?

A: 1. Install EGit plugin.
2. Go to Window → Preferences → Team → Git to set Git configurations.
3. Use Git Repositories view to clone, push, pull and manage Git projects.

- Q: Can Eclipse be used for Jenkins pipeline development?

A: Yes. Eclipse can be used to write and manage Jenkinsfiles for pipeline as code. You can also use the Jenkins plugin for Eclipse to monitor jobs and builds directly.

- Q: How do you use Maven in Eclipse for DevOps?

A: - Right-click on project → Configure → Convert to Maven Project.
- Use pom.xml to define dependencies and build configurations.
- Maven helps with CI/CD, testing, and packaging.

- Q: What is the role of Eclipse in CI/CD workflows?

A: Eclipse is used to develop code, manage version control, run tests, package applications (using Maven/Gradle), and integrate with CI tools like Jenkins.

- Q: How to use Docker from Eclipse?

A: - Install Docker Tooling plugin.
- Connect Docker Daemon.
- Use Eclipse Docker view to build images, run containers, and manage volumes/networks.

- Q: What version of Java should be used with Eclipse for DevOps projects?

A: It depends on the project, but typically Java 8 or above is preferred for compatibility with tools like Jenkins and Spring Boot.

- Q: How can Eclipse help in automation testing for DevOps?

A: Eclipse supports automation testing frameworks like JUnit, TestNG, and integration with Selenium, which are key in DevOps testing pipelines.

Viva Questions and Answers – Jenkins Setup on AWS in DevOps

- Q: What is Jenkins and why is it used in DevOps?

A: Jenkins is an open-source automation server that is used to automate parts of software development such as building, testing, and deploying. It plays a critical role in CI/CD pipelines.

- Q: What are the steps to set up Jenkins on AWS?

A: 1. Launch an EC2 instance (Ubuntu preferred).
2. Install Java (required by Jenkins).
3. Add Jenkins repository and install Jenkins.
4. Start Jenkins and open the web interface using the public IP with port 8080.
5. Retrieve the initial admin password and install recommended plugins.

- Q: What security group settings are required for Jenkins on AWS?

A: You need to allow inbound traffic on port 8080 (Jenkins default port) and optionally SSH (port 22) for remote access.

- Q: How do you access Jenkins installed on an EC2 instance?

A: Use the public IP of the EC2 instance and access Jenkins in the browser with: http://<ec2-public-ip>:8080

- Q: How do you install Jenkins using the command line on Ubuntu?

A: Use the following commands:
- sudo apt update
- sudo apt install openjdk-11-jdk
- wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
- sudo sh -c 'echo deb https://pkg.jenkins.io/debian binary/ > /etc/apt/sources.list.d/jenkins.list'
- sudo apt update
- sudo apt install jenkins

- Q: Where is the initial admin password for Jenkins stored after installation?

A: It is stored in: /var/lib/jenkins/secrets/initialAdminPassword

- Q: What are Jenkins pipelines?

A: Jenkins pipelines are a suite of plugins that support implementing and integrating continuous delivery pipelines into Jenkins using pipeline as code.

- Q: Can you integrate Jenkins with AWS CodeDeploy or S3?

A: Yes, Jenkins can be integrated with AWS services like CodeDeploy, S3, EC2 using plugins and custom shell scripts for deployment.

- Q: What plugins are useful for Jenkins in a DevOps setup?

A: - Git plugin
- Maven plugin
- Pipeline plugin
- SSH plugin
- AWS CodeDeploy plugin
- Docker plugin

- Q: What are the best practices for securing Jenkins on AWS?

A: - Enable HTTPS access
- Use IAM roles and policies for AWS access
- Limit access using security groups
- Configure Jenkins user roles and access control

DevOps Viva Questions: Building WAR Files

Basic Concepts

What is a WAR file?

A WAR (Web Application Archive) file packages all the components of a web application (like JSPs, Servlets, classes, and static files) and is deployed to a servlet container like Tomcat.

How do you build a WAR file using Maven?

You can build a WAR file by using `mvn clean package`. Ensure your `pom.xml` has <packaging>war</packaging>.

What is the structure of a WAR file?

Typical structure:
/WEB-INF/
  web.xml
  /classes/
  /lib/
index.jsp

Build Tools & CI/CD

Which tools do you use to build WAR files in your DevOps pipeline?

Common tools: Maven, Gradle, Jenkins, GitLab CI, Bamboo.

How do you automate WAR file creation in Jenkins?

Set up a Jenkins job with:
- Source code checkout
- Maven build step: mvn clean package
- Archive the WAR file as a post-build action

Where is the WAR file located after a Maven build?

In the target/ directory of the project root.

Deployment & Integration

How do you deploy a WAR file to Tomcat using Jenkins?

Use Jenkins plugins like 'Deploy to container' or use SCP/SFTP to copy the WAR to Tomcat's webapps/ directory and restart the server if needed.

How do you handle versioning for WAR files?

By configuring the version in pom.xml (<version> tag), e.g., myapp-1.0.0.war.

Configuration Management

How do you manage different environment configurations in WAR builds?

Use Maven profiles or externalize configs (e.g., application.properties, environment variables).

Can you build different WAR files for dev, test, and prod?

Yes, using Maven profiles and build parameters to generate environment-specific WARs.

Error Handling

What do you do if the WAR build fails in Jenkins?

Check the console output/logs, review Maven errors, fix code or dependency issues, and retry.

How do you ensure the WAR file is built correctly before deployment?

Run unit tests, integration tests, and use Jenkins build status checks or quality gates (e.g., SonarQube).

Advanced & Best Practices

What are best practices for building and deploying WAR files in DevOps?

Automate builds and deployments
Use artifact repositories (e.g., Nexus, Artifactory)
Apply consistent versioning
Use CI/CD with quality checks

How can Docker be used in WAR-based application deployment?

Build a Docker image with Tomcat and copy the WAR file inside the image. Use it in containerized deployments.

DevOps Viva Questions: Ansible Setup and SSH Keys

Ansible Setup

What is Ansible?

Ansible is an open-source automation tool used for configuration management, application deployment, and task automation.

How do you install Ansible?

On most Linux distributions: sudo apt install ansible or sudo yum install ansible.

What is an inventory file in Ansible?

An inventory file contains the list of hosts (or groups of hosts) that Ansible manages.

What is the default location of the Ansible configuration file?

The default location is /etc/ansible/ansible.cfg.

How do you check the Ansible version?

Run the command: ansible --version

What are playbooks in Ansible?

Playbooks are YAML files where Ansible automation is defined through tasks.

How do you test Ansible connectivity to managed nodes?

Use the ping module: ansible all -m ping

SSH Keys in Ansible

Why are SSH keys important in Ansible?

Ansible connects to managed nodes over SSH. SSH keys enable passwordless, secure communication.

How do you generate an SSH key?

Use the command: ssh-keygen

How do you copy the public SSH key to a remote server?

Use the command: ssh-copy-id user@remote_host

Where are SSH keys typically stored?

In the ~/.ssh/ directory. The private key is id_rsa and the public key is id_rsa.pub.

How do you specify a custom SSH key in Ansible?

You can define it in the inventory file or use the --private-key option:
ansible all -m ping --private-key=~/.ssh/my_key

What if a host is prompting for SSH confirmation in Ansible?

Ensure the host is in the known_hosts file or use the parameter:
ansible_ssh_common_args='-o StrictHostKeyChecking=no'

How do you use SSH agent forwarding with Ansible?

Enable SSH agent forwarding and load your key using ssh-add. Use 'forward_agent=yes' in the inventory or SSH config.

DevOps Viva Questions: Deploying Artifact on the Test Server

Basic Deployment Concepts

What is an artifact in DevOps?

An artifact is a file or set of files (e.g., JAR, WAR, ZIP) that are the output of the build process and are ready for deployment.

What tools are used for artifact deployment?

Common tools include Jenkins, Ansible, Docker, Kubernetes, and artifact repositories like Nexus or Artifactory.

What is a test server in CI/CD?

A test server is an environment used to deploy and test application artifacts before moving to staging or production.

Artifact Deployment Process

How do you deploy a WAR file to a test server manually?

You can SCP the WAR file to the server and place it in the appropriate directory (e.g., Tomcat's webapps) and restart the server.

How do you automate deployment of artifacts using Jenkins?

By creating a Jenkins pipeline or freestyle job that pulls the artifact from the build or repository and deploys it via SSH, SCP, or tools like Ansible.

What plugins in Jenkins help with deployment?

Deploy to container plugin, SSH plugin, Copy Artifact plugin, and Pipeline plugin.

Integration with Repositories

How do you pull an artifact from Nexus or Artifactory?

Use a script or Maven/Gradle to download the artifact using repository URL and credentials.

What are best practices for naming artifacts?

Include project name, version, and environment in the artifact name (e.g., myapp-1.0.0-test.war).

Verification and Validation

How do you ensure that the artifact was deployed correctly?

Check application logs, run smoke tests, or monitor endpoints to verify the application is running as expected.

What steps do you take if deployment fails on the test server?

Check logs, verify SSH access, ensure enough disk space, confirm that dependencies are installed, and roll back if needed.

Advanced Deployment Strategies

How do you use Ansible for deploying artifacts?

Write a playbook to copy the artifact to the test server and restart the necessary services.

What is blue-green deployment?

A technique where two environments (blue and green) are maintained, allowing zero-downtime deployment by switching traffic between them.

Can Docker be used to deploy artifacts?

Yes, by packaging the application into a Docker image and running it on a container on the test server.

Jenkins Automation Viva Questions and Answers

1. What is Jenkins and why is it used for automation?

Jenkins is an open-source automation server used to automate the building, testing, and deployment of software. It supports continuous integration and continuous delivery (CI/CD).

2. Explain how Jenkins fits into the CI/CD pipeline.

Jenkins acts as a central hub in the CI/CD pipeline by automatically building and testing code changes and deploying them to production environments.

3. What are the key features of Jenkins that support automation?

Key features include pipeline support, plugin extensibility, distributed builds, integration with version control systems, and scheduling via cron syntax.

4. What is a Jenkins job or project?

A Jenkins job is a task or a set of tasks configured in Jenkins to perform actions like building code, running tests, or deploying applications.

5. What are the types of Jenkins jobs?

Types include Freestyle project, Pipeline project, Multi-configuration project, Folder, and Multibranch Pipeline.

6. What is the difference between a freestyle project and a pipeline project?

A freestyle project is a simple and easy-to-configure job, whereas a pipeline project allows complex build processes using code defined in a Jenkinsfile.

7. How can Jenkins automate the build process?

Jenkins can pull code from a repository and run build tools like Maven, Gradle, or Ant automatically on each commit.

8. How does Jenkins automate testing?

Jenkins can run automated tests like unit tests or integration tests using tools like JUnit, TestNG, or Selenium after every build.

9. Explain how to automate deployment using Jenkins.

Jenkins can use shell scripts or plugins to deploy built applications to staging or production servers automatically.

10. How do you trigger a Jenkins job automatically on code push?

By configuring webhooks in GitHub/GitLab to notify Jenkins of code changes, or using the Poll SCM option in Jenkins.

11. What are post-build actions in Jenkins and how do they help in automation?

Post-build actions are steps that occur after the main build, like archiving artifacts, sending notifications, or triggering other jobs.

12. What are Jenkins plugins? Can you name a few useful plugins for automation?

Plugins extend Jenkins functionality. Examples: Git Plugin, Pipeline Plugin, Maven Plugin, Docker Plugin, Slack Notification Plugin.

13. How can Jenkins be integrated with Git/GitHub?

By installing the Git plugin and configuring the repository URL, credentials, and webhooks in GitHub.

14. Explain how Jenkins integrates with Docker or Kubernetes for automation.

Jenkins can build Docker images and run containers using the Docker plugin. Kubernetes plugin allows Jenkins to dynamically provision build agents.

15. What role does Maven or Gradle play in Jenkins automation?

Maven and Gradle are build tools that Jenkins can invoke to compile code, run tests, and package applications.

16. What is a Jenkins Pipeline? How is it used for automation?

A Jenkins Pipeline is a suite of plugins that supports implementing and integrating continuous delivery pipelines as code.

17. What is a Declarative vs Scripted Pipeline in Jenkins?

Declarative Pipeline uses a more structured and simpler syntax. Scripted Pipeline is more flexible and written in Groovy.

18. How do you use Jenkinsfile for automation?

A Jenkinsfile contains the pipeline code and is stored in the version control repository, allowing Jenkins to automate tasks consistently.

19. Can you automate notifications (email/Slack) using Jenkins?

Yes, Jenkins supports sending notifications through plugins like Email Extension or Slack Notification.

20. How does Jenkins support parallel and sequential job execution?

Pipelines allow defining parallel stages and sequential steps to manage complex workflows efficiently.

21. What are some best practices for Jenkins job automation?

Use version control for Jenkinsfiles, keep jobs modular, monitor builds, secure credentials, and clean up old jobs.

22. How do you secure sensitive information like passwords in automated Jenkins jobs?

Use Jenkins Credentials plugin to securely store and inject secrets into jobs.

23. How do you manage credentials securely in Jenkins?

Store them in Jenkins Credential Manager and access them using environment variables or withCredentials block in pipelines.

24. What would you do if a Jenkins job fails during automation?

Check console logs, debug the issue, verify dependencies, and check integration points like Git or build tools.

25. How can you troubleshoot a failing pipeline?

Review Jenkins logs, use echo/debug statements, check for script syntax errors, and validate plugin configurations.

26. How do you monitor Jenkins job performance and logs?

Use the Jenkins dashboard, console output, and plugins like Monitoring or Log Parser to analyze job performance.

Build and Deploy Selenium Grid for Chrome and Firefox Testing in DevOps - Viva Q&A

1. What is the role of Selenium Grid in a DevOps pipeline?

Selenium Grid enables automated parallel testing across multiple browsers and environments, improving speed and reliability in CI/CD pipelines.

2. How can Selenium Grid be deployed in a DevOps environment?

Selenium Grid can be deployed using Docker, Kubernetes, or virtual machines, often integrated with Jenkins, GitLab CI, or other CI/CD tools.

3. What is the benefit of using Docker to deploy Selenium Grid?

Docker simplifies setup, ensures environment consistency, and enables easy scaling of nodes for Chrome and Firefox browsers.

4. What is a Docker Compose file for Selenium Grid?

A Docker Compose file defines services such as the Selenium Hub, Chrome Node, and Firefox Node, allowing simultaneous deployment with one command.

5. How would you define Chrome and Firefox nodes in Docker Compose?

Example:

```
selenium-node-chrome:
  image: selenium/node-chrome
  environment:
    - HUB_HOST=selenium-hub
selenium-node-firefox:
  image: selenium/node-firefox
  environment:
    - HUB_HOST=selenium-hub
```

6. How is Jenkins used in automating Selenium Grid tests?

Jenkins can trigger jobs that execute tests on Selenium Grid as part of the CI pipeline. Tests use RemoteWebDriver configured with the Grid URL.

7. How do you write a Jenkinsfile for running tests on Selenium Grid?

Use `pipeline` syntax to define stages like 'Build', 'Deploy Grid (Docker)', 'Run Tests', and 'Tear Down'. Include Docker or shell commands to start and stop containers.

8. What are the common issues in Selenium Grid deployment and how do you troubleshoot them?

Issues include network connectivity between Hub and Nodes, version mismatches, or container startup failures. Logs and port configurations are essential for troubleshooting.

9. How do you scale the Selenium Grid for heavy test loads?

Add more browser nodes (containers or VMs), use dynamic scaling with Kubernetes, or auto-scale based on demand using orchestration tools.

10. What are the advantages of Selenium Grid 4 in DevOps over older versions?

Selenium Grid 4 offers better support for Docker and Kubernetes, built-in observability, and simplified configuration using event-driven architecture.

Viva Questions and Answers – Docker Image Creation and Push to Docker Hub

What is Docker?

Docker is an open-source platform that automates the deployment, scaling, and management of applications using containerization. Containers package the application code along with its dependencies to run consistently across environments.

What is a Dockerfile?

A Dockerfile is a text file that contains a set of instructions to build a Docker image. Each instruction represents a command executed in the image creation process.

What is the purpose of Docker Hub?

Docker Hub is a cloud-based repository where Docker users can store, share, and manage Docker images.

How do you create a Docker image for an application?

1. Write a Dockerfile with the application's environment and setup instructions.
2. Use the command `docker build -t image_name .` to build the image.

How do you push a Docker image to Docker Hub?

1. Login to Docker Hub using `docker login`.
2. Tag the image using `docker tag image_name username/repository:tag`.
3. Push it using `docker push username/repository:tag`.

What command is used to run a Docker container from an image?

`docker run -d -p 8080:80 image_name`

What is the difference between a container and an image?

- An image is a blueprint for creating containers.
- A container is a running instance of an image.

What is the use of EXPOSE in Dockerfile?

The EXPOSE instruction informs Docker that the container listens on the specified network ports at runtime.

What is the difference between CMD and ENTRYPOINT?

- CMD sets default command and parameters.
- ENTRYPOINT is used to define a fixed command to run.

How can you automate Docker image creation in a DevOps pipeline?

Using CI/CD tools like Jenkins, GitHub Actions, or GitLab CI to build and push Docker images after code is pushed or merged.

Sample Dockerfile: Node.js App

```
# Use base image
FROM node:18

# Set working directory
WORKDIR /app

# Copy package files and install dependencies
COPY package*.json ./
RUN npm install

# Copy rest of the app files
COPY . .

# Expose port
EXPOSE 3000

# Run the app
CMD ["node", "index.js"]
```

Commands to Build and Push

```
docker build -t my-node-app .
docker tag my-node-app your_dockerhub_username/my-node-app:latest
docker login
docker push your_dockerhub_username/my-node-app:latest
```

6.Deploy the artifact on the Test Server?

1. What is an artifact in DevOps?

Answer:
An artifact is a compiled or built file, such as a JAR, WAR, binary, or Docker image, created during the build stage of the software development lifecycle and deployed to environments like testing, staging, or production.

2. How do you deploy an artifact to a test server?

Answer:
Deployment can be done via CI/CD tools (like Jenkins, GitLab CI, GitHub Actions) using SSH, SCP, FTP, or Docker/Kubernetes. Example using scp:

```
scp myapp.war user@test-server:/opt/tomcat/webapps/
```

3. Which tools are commonly used to deploy artifacts?

Answer:

- Jenkins
- Ansible
- GitHub Actions
- GitLab CI/CD
- Docker + Kubernetes
- Nexus/Artifactory (for storing)

4. What is a test environment in DevOps?

Answer:
A test environment mimics production where new builds or features are tested before release. It often includes servers, databases, and services similar to production.

5. What is the role of Jenkins in deploying artifacts?

Answer:
Jenkins automates deployment by pulling the latest build, copying the artifact to the server, and restarting services using a pipeline job.

6. How do you ensure the test server is clean before deploying?

Answer:

- Stop running services

- Remove old artifacts
- Clear temp or cache directories
- Use automation (like Ansible or Jenkins scripts)

7. What is rollback and how is it done?

Answer:
Rollback is reverting to a previous stable version if the current deployment fails. It's done by storing previous artifacts or using Docker tags/snapshots.

8. How do you validate a successful deployment?

Answer:

- Check logs
- Verify running processes
- Send HTTP requests to endpoints
- Use CI tools for post-deployment tests

9. How does environment-specific configuration work?

Answer:
Environment variables, config files, or tools like Spring profiles or Ansible inventory files are used to load settings specific to test or production environments.

10. What challenges do you face during deployment on a test server?

Answer:

- Environment inconsistencies
- Port conflicts
- Configuration mismatches
- Insufficient permissions
- Dependency issues

---

7. Create deployment resource using Kubernetes ?

1. What is a Deployment in Kubernetes?

Answer:
A Deployment in Kubernetes is a controller that provides declarative updates for Pods and ReplicaSets. It allows you to define the desired state for your application and the Deployment Controller will manage the state for you.

2. What command is used to apply a Deployment YAML file?

Answer:

```
kubectl apply -f deployment.yaml
```

3. What is the purpose of the replicas field in a Deployment?

Answer:
The replicas field specifies the number of pod replicas that should be running at any given time. Kubernetes ensures that this number is maintained by starting new pods or terminating existing ones.

4. How can you update an application using Deployment?

Answer:
You can update the container image or other configuration in the Deployment YAML and reapply it using kubectl apply -f. Kubernetes will perform a rolling update without downtime.

5. How do you roll back a Deployment?

Answer:

```
kubectl rollout undo deployment/my-app
```

6. What is the difference between a Deployment and a ReplicaSet?

Answer:
A ReplicaSet ensures that a specified number of pod replicas are running, but a Deployment provides declarative updates and rollback capabilities on top of ReplicaSets.

7. How do you check the status of a Deployment?

Answer:

```
kubectl get deployments
kubectl describe deployment my-app
```

8. What happens if one of the pods in a Deployment fails?

Answer:
Kubernetes will automatically restart or recreate the pod to maintain the desired state defined in the Deployment.

9. Can a Deployment manage multiple containers in one pod?

Answer:
Yes. You can define multiple containers in the spec.template.spec.containers list within the Deployment configuration.

11.Setup Grafana for Devops ?

1. What is Grafana?

Grafana is an open-source data visualization tool used for monitoring, alerting, and analyzing metrics from various data sources, including Prometheus.

2. What are the main features of Grafana?

- Interactive and customizable dashboards
- Multiple data source support (Prometheus, InfluxDB, Elasticsearch, etc.)
- Alerting and notification system
- Role-based access control

3. How do you install Grafana?

- Using apt:

sudo apt-get install -y grafana

- Using Docker:

docker run -d -p 3000:3000 grafana/grafana

- Access it at http://localhost:3000

4. What are Grafana dashboards?

Dashboards in Grafana are collections of visual panels that display metrics and logs in real-time.

5. How do you connect Grafana to Prometheus?

- Add Prometheus as a data source.
- Enter Prometheus URL (e.g., http://localhost:9090).
- Save and test the connection.

6. How do you create a dashboard in Grafana?

- Navigate to 'Create' → 'Dashboard'
- Add a new panel

- Write a PromQL query to fetch data from Prometheus
- Choose visualization type (graph, gauge, table, etc.)

7. How do you set up alerts in Grafana?

- In a panel, go to the 'Alert' tab
- Set conditions, evaluation intervals
- Configure notification channels (Slack, email, etc.)

8. What are panels in Grafana?

Panels are the visual representations (graphs, tables, etc.) within a dashboard used to show metrics data.

9. How is Grafana useful in DevOps?

- Visualize CI/CD performance
- Monitor infrastructure health
- Real-time dashboard for deployment events and metrics
- Centralized view for logs, traces, and metrics

10. Can Grafana be integrated with alerting tools?

Yes. Grafana supports alerting integration with:

- Slack
- Email
- PagerDuty
- Microsoft Teams
- Webhooks

---

12. Setup Prometheus for Devops ?

1. Introduction:

- **Prometheus** is an open-source systems monitoring and alerting toolkit, particularly well-suited for monitoring dynamic, cloud-native environments such as Kubernetes. It uses a pull-based model to scrape metrics from configured endpoints.

2. Key Concepts:

- **Metrics:** Data points collected over time, usually in the form of time series.
- **PromQL:** Prometheus Query Language used to query the collected metrics.
- **Exporters:** Components that expose metrics in a format that Prometheus can scrape.
- **Alertmanager:** Manages alerts generated by Prometheus.

3. Installation:

- **Running Prometheus:**

```
wget
https://github.com/prometheus/prometheus/releases/download/v2.30.0/p
rometheus-2.30.0.linux-amd64.tar.gz
tar xvfz prometheus-*.tar.gz
cd prometheus-*
./prometheus --config.file=prometheus.yml
```

Docker:

```
docker run -p 9090:9090 prom/prometheus
```

4. Prometheus Configuration:

Basic prometheus.yml Configuration:

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
```

Adding Targets:

```
- job_name: 'node_exporter'
  static_configs:
    - targets: ['localhost:9100']
```

5. Prometheus Query Language (PromQL):

Basic Queries:

```
up
rate(http_requests_total[5m])
```

Aggregations:

```
sum(rate(http_requests_total[5m]))
avg_over_time(http_requests_total[5m])
```

Recording Rules:

```
groups:
- name: example
```

```
  rules:
  - record: job:http_inprogress_requests:sum
    expr: sum(http_inprogress_requests) by (job)
```

6. Exporters:

Node Exporter: Collects system-level metrics.

```
wget
https://github.com/prometheus/node_exporter/releases/download/v1.2.2
/node_exporter-1.2.2.linux-amd64.tar.gz
tar xvfz node_exporter-*.tar.gz
./node_exporter
```

Custom Exporter: Writing a custom exporter using Python.

```
from prometheus_client import start_http_server, Gauge
import random
import time

g = Gauge('random_number', 'A random number')

def generate_random_number():
    while True:
        g.set(random.random())
        time.sleep(5)

if __name__ == '__main__':
    start_http_server(8000)
    generate_random_number()
```

7. Alerts and Alertmanager:

Alerting Rules:

```
groups:
- name: example
  rules:
  - alert: HighMemoryUsage
    expr: node_memory_Active_bytes / node_memory_MemTotal_bytes *
100 > 90
    for: 5m
    labels:
      severity: critical
    annotations:
      summary: "High memory usage detected on {{ $labels.instance
}}"
      description: "Memory usage is above 90% for more than 5
minutes."
```

Alertmanager Configuration:

```
global:
  resolve_timeout: 5m
```

```
route:
  group_by: ['alertname']
  receiver: 'email'

receivers:
- name: 'email'
  email_configs:
  - to: 'your-email@example.com'
    from: 'prometheus@example.com'
    smarthost: 'smtp.example.com:587'
    auth_username: 'username'
    auth_password: 'password'
```

8. Prometheus Federation:

Setting Up Federation:

```
scrape_configs:
- job_name: 'federate'
  honor_labels: true
  metrics_path: '/federate'
  params:
    match[]:
      - '{job="prometheus"}'
  static_configs:
    - targets:
      - 'prometheus-server-1:9090'
      - 'prometheus-server-2:9090'
```

9. Monitoring Kubernetes with Prometheus:

Deploying Prometheus on Kubernetes:

```
apiVersion: monitoring.coreos.com/v1
kind: Prometheus
metadata:
  name: prometheus
spec:
  replicas: 1
  serviceAccountName: prometheus
  serviceMonitorSelector:
    matchLabels:
      team: frontend
  resources:
    requests:
      memory: 400Mi
  storage:
    volumeClaimTemplate:
      spec:
        storageClassName: standard
        resources:
```

```
          requests:
            storage: 50Gi
```

ServiceMonitor Example:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: example-monitor
spec:
  selector:
    matchLabels:
      app: example
  endpoints:
    - port: web
```

10. Advanced Prometheus Concepts:

- **Thanos:** Extends Prometheus with long-term storage, global querying, and downsampling.
- **Cortex:** Multi-tenant, horizontally scalable Prometheus as a service.

11. Prometheus Security:

- **Basic Authentication:**

```
basic_auth:
  username: admin
  password: admin
```

TLS/SSL Configuration:

```
tls_config:
  ca_file: /etc/prometheus/certs/ca.crt
  cert_file: /etc/prometheus/certs/prometheus.crt
  key_file: /etc/prometheus/certs/prometheus.key
```

12. Troubleshooting Prometheus:

- **Common Issues:**
    - **High Cardinality Metrics:** Too many unique time series can overwhelm Prometheus.
    - **Slow Queries:** Optimize queries by avoiding high cardinality and using efficient aggregations.
- **Debugging:**
    - Use the **promtool** command-line tool to check configuration files.
    - **Prometheus UI** provides an interface to debug queries and examine time series data.