Introduction to SRE

Practical work 1- Setting Up an SRE Environment

Task 1. Setting Up an SRE Environment

Task 1.1. Installation of a virtualization platform or containerization environment:

- Research and choose a suitable platform (VMware, Hyper-V, Docker, Kubernetes, etc.) based on project requirements.
- Download and install the chosen platform, ensuring system compatibility.
- Create and run a sample virtual machine or container to ensure the installation is successful.

Tasks for Setting Up Different Platforms:

1. VMware:

Research and Choose:

- Understand the features and benefits of VMware's various products, like VMware Workstation, VMware Fusion, and VMware vSphere.
- Assess the system requirements for the chosen VMware product.

Installation:

- Download the appropriate version of the VMware product.
- Follow the installation wizard, ensuring you allocate enough resources (RAM, CPU) for the virtual machines you plan to run.
- Install VMware tools for enhanced performance and usability.

Sample Virtual Machine:

- Create a new virtual machine using the VMware interface.
- Install an OS on the virtual machine, like Windows or Linux, and ensure it boots up properly.

2. Hyper-V:

Research and Choose:

- Familiarize yourself with Hyper-V's capabilities, especially in comparison with other hypervisors.
- Ensure your system runs Windows 10 Professional, Enterprise, or Education editions or Windows Server.

Installation:

- Activate Hyper-V feature through Windows Features or via PowerShell.
- After installation, reboot your machine and open the Hyper-V Manager.

Sample Virtual Machine:

- In Hyper-V Manager, create a new virtual machine.

- Install a guest OS and ensure you can interact with it through the Hyper-V console.

3. Docker:

Research and Choose:
- Understand the basic principles of containerization and how Docker differs from traditional virtualization.
- Assess whether Docker Desktop or Docker Server is appropriate for your needs.

Installation:
- Download Docker from the official website.
- Follow the installation instructions specific to your OS (Windows, macOS, or Linux).
- Start Docker and ensure the daemon runs properly.

Sample Container:
- Run a "Hello World" container using docker run hello-world to ensure Docker can pull and run containers.

4. Kubernetes:
Research and Choose:
- Learn about Kubernetes' container orchestration capabilities and its difference from Docker Swarm or other orchestrators.
- Decide if you want a local setup (like Minikube) or if you're going to use a cloud provider's Kubernetes service.

Installation:
- For a local setup, install Minikube and start a local Kubernetes cluster.
- If opting for a cloud provider, set up a Kubernetes cluster using their specific instructions (like Google Kubernetes Engine or Amazon EKS).

Sample Pod:
- Deploy a simple application using kubectl to ensure your Kubernetes cluster can schedule and run pods.

Note. For each platform, after performing the tasks, it's essential to document each step, any challenges faced, and the solutions implemented.

Requirements for Students:

System Requirements:
- A computer system with a minimum of 8 GB RAM (16 GB recommended for better performance).
- At least 100 GB of free disk space.
- A 64-bit processor with virtualization capabilities enabled in BIOS.
- Internet connectivity for software downloads and online resources.

Software & Licenses:

- Appropriate licenses or access to free versions of the virtualization/containerization platforms being studied (e.g., VMware Workstation Player, Docker Desktop, Minikube).
- Operating system ISO images or installation files for creating virtual machines.
- Backup software or tools compatible with the chosen platform.
- Basic Technical Know-how:
- Familiarity with their operating system's terminal or command-line interface.
- A foundational understanding of virtualization and containerization concepts.
- Basic knowledge of networking concepts like IP addresses, DNS, and ports.

Documentation Tools:

- Prepare a report with screenshots of completed tasks.
- Access to documentation tools/software such as Microsoft Word, Google Docs, or any note-taking app.
- Screen capturing tools for taking screenshots during setup and configuration.
- Diagramming tools like Draw.io, Lucidchart, or Microsoft Visio for creating network or infrastructure diagrams, if necessary.

Safety and Best Practices:

- Ensure data backup of their primary systems to prevent any data loss during software installations.
- Utilize sandboxed environments or isolated networks during configurations to prevent unintentional disruptions to primary network settings.
- Always follow best practices and guidelines provided in official documentation when setting up environments.

Task 1.2. Configuration of network settings, storage, and backup mechanisms:

- Configure the network settings to allow the virtual machines or containers to communicate with the host and other VMs/containers.

- Allocate and manage storage efficiently to cater to the needs of different virtual machines or containers.

- Set up a backup mechanism, considering aspects like frequency, backup medium, retention period, and recovery process.

Objective: To understand and configure the underlying infrastructure that supports SRE operations, ensuring connectivity, data integrity, and recovery capabilities.

Network Configuration:

Virtual Machines:

- Bridge Mode: Allows the VM to connect to the network as if it were a physical device. The VM will get its IP address from the network's DHCP server.
- NAT Mode: The VM communicates through the host machine, useful for internet access.
- Host-Only Mode: Allows communication only between VMs and host machines.
- Firewall & Security: Ensure ports required by applications are open, and unnecessary ports are closed or restricted.

Containers (e.g., Docker):

- Bridge Network: Default Docker network, allows containers to communicate with each other and the host.
- Host Network: Containers share the network namespace of the host.
- Overlay Network: For distributed applications across multiple hosts or clusters.
- Macvlan Network: Assign a MAC address to a container, making it appear as a physical network device.

Storage Configuration:

Virtual Machines:

- Dynamic Allocation: Start with minimal storage and expand as needed.
- Fixed Size: Allocates the entire space at creation; generally offers better performance.
- Shared Folders: Allows sharing files between host and VM.

Containers:

- Volumes: Persistent storage outside the container's lifecycle. Useful for databases or logs.
- Bind Mounts: Map a host file or directory to a container file or directory.
- tmpfs Mounts: Temporary storage only available to the container and deleted on container exit.

Backup Mechanisms:

Virtual Machines:

- Snapshots: Capture the state, data, and settings of a VM. Useful for quick rollbacks.
- Cloning: Create a full copy of the VM.
- Export/Import: Save the VM as an OVA or OVF for portability.

Containers:

- Commit: Create a new image from the container's changes.
- Docker Save/Load: Allows you to save a Docker image as a tar archive and then load it back.
- General Backup Considerations:
- Frequency: Decide on full backups vs. incremental or differential backups.
- Medium: On-site storage, cloud storage, or off-site tapes.
- Retention Period: How long backups are stored based on compliance and business needs.
- Recovery Process: Regularly test backup restoration to ensure data integrity and availability.

Requirements for Students:

1. Understand Basic Networking: Familiarity with IP addresses, subnets, gateways, and routing.
2. Virtualization & Container Tools: Experience with the chosen virtualization (like VMware, VirtualBox) or containerization (Docker, Kubernetes) tools.
3. Backup Knowledge: Basic understanding of backup methodologies and tools.

4. Documentation Skills: Ability to document configurations, changes, and backup steps clearly.

5. Research Acumen: Should be able to refer to tool-specific documentation for any advanced configurations or troubleshooting.

6. Safety & Etiquette: Ensure no disruption to existing network setups and always backup data before making significant changes.

7. Hands-on Lab/Practical Setup: Access to a sandbox environment where they can freely configure and test without disrupting actual services.

8. Prepare a report with screenshots of completed tasks.

2. Introduction to SRE tools

The efficiency of an SRE's work largely depends on the toolset at their disposal. This section will explore the installation and functional overview of some quintessential SRE tools, highlighting their role in monitoring, logging, and alerting.

Tasks 2.1. Installation of essential SRE tools

- Identify a list of tools essential for the SRE role, including monitoring (e.g., Prometheus), logging (e.g., ELK Stack), and alerting (e.g., Alertmanager).
- Install the identified tools, ensuring compatibility and addressing any dependencies.
- Validate the installation by running basic commands or accessing the tool's dashboard.

Prometheus Installation Guide

Objective: To install Prometheus for monitoring system performance.

a. Tool Selection and Purpose:

Prometheus https://prometheus.io : An open-source system monitoring and alerting toolkit.

b. Installation Process:

Pre-installation:

1. System Requirements:
- OS: Linux (Ubuntu, CentOS, or any other distribution), MacOS, or Windows.
- CPU: Minimum 2 CPUs for production environments.
- RAM: Minimum 1GB, but 2GB or more is recommended.
- Storage: Varies based on usage, but 20GB is a reasonable starting point.

2. Prerequisites:
- Ensure you have `curl` or `wget` installed to fetch the binaries.
- User with `sudo` permissions (if installing system-wide).

Installation:

1. Downloading Prometheus:
- Go to the [official Prometheus downloads page](https://prometheus.io/download/).
- Choose the appropriate version for your OS and architecture.
- Download using a browser or use `curl` or `wget`. For example:

```bash
wget https://github.com/prometheus/prometheus/releases/download/vX.X.X/prometheus-X.X.X.linux-amd64.tar.gz
```

2. Extracting Prometheus:
- Extract the downloaded tar.gz file.
```bash
tar xvzf prometheus-X.X.X.linux-amd64.tar.gz
```

3. Navigate to the Prometheus directory:
```bash
cd prometheus-X.X.X.linux-amd64
```

4. Run Prometheus:
- You can run Prometheus with the default configuration using:
```bash
./prometheus
```

Post-installation:

1. Verification:
- Open your browser and navigate to `http://localhost:9090/`.
- You should see the Prometheus web UI with graphs, alerts, and other options.

2. Screenshots:
- Capture screenshots of the running Prometheus UI.
- Document any initial configurations or settings.

Note: *The above guide is a basic installation of Prometheus for quick setup. In a real-world scenario, you'd likely want to set up Prometheus as a system service, ensure it starts on boot, and use a configuration management tool to manage its configuration. Similar steps can be created for the ELK Stack and Alertmanager. Ensure that after every major step, screenshots are taken to document the process visually.*

Task 2.2. Overview and Demonstration of Tool Functionalities

- For each tool, outline its primary purpose and how it aids the SRE function.
- Demonstrate key features of the tool, such as setting up a monitor in Prometheus or creating an alert rule in Alertmanager.
- Execute basic troubleshooting operations, like querying logs in ELK or analyzing metrics in Grafana.

1. **Prometheus https://prometheus.io**

Primary Purpose: Prometheus is an open-source monitoring and alerting toolkit primarily designed for reliability and scalability. It gathers multidimensional data metrics from varied sources and stores them as time series data.

How it aids the SRE function:

- Proactive Monitoring: With Prometheus, SRE teams can get ahead of issues before they affect users. It allows them to monitor application health and other technical indicators.
- Scalability: Prometheus can scale with your applications, capturing millions of metrics from different endpoints.
- Flexibility: Allows querying data metrics with PromQL, a flexible query language.

Key Feature Demonstration:

- Setting up a Monitor: Demonstrate how to set up targets (like application endpoints or hosts) for Prometheus to scrape data from. Show how to define scrape intervals and set up basic authentication if necessary.
- Using the Expression Browser: Show how to use the built-in expression browser to run queries against the Prometheus database.

Troubleshooting:

- Show how to use PromQL to query metrics and identify anomalies. For example, if the rate of HTTP 500 errors spikes, it can be detected using a PromQL query.

2. Alertmanager https://prometheus.io/docs/alerting/latest/alertmanager/

Primary Purpose: Alertmanager handles alerts sent by client applications like Prometheus. It takes care of deduplicating, grouping, and routing them to the correct receiver integrations like email, PagerDuty, etc.

How it aids the SRE function:

- Incident Management: By efficiently routing alerts to the right team or individual, it ensures rapid response to incidents.
- Silencing & Inhibition: Alertmanager can silence specific alerts and inhibit alerts based on their labels, reducing noise.

Key Feature Demonstration:

- Creating an Alert Rule: Show how to define alerting rules in Prometheus and then route them to Alertmanager. Demonstrate setting conditions for an alert to trigger.
- Setting up Routes: Demonstrate how to route alerts to different receivers based on their severity or the service they relate to.

Troubleshooting:

- Simulate a scenario where a predefined alert threshold is crossed, triggering an alert, and see how Alertmanager handles and routes it.

3. ELK Stack (Elasticsearch, Logstash, Kibana)

1. **Elasticsearch:** A distributed, RESTful search and analytics engine.

2. **Logstash:** A server-side data processing pipeline that ingests data from multiple sources, transforms it, and sends it to a "stash" like Elasticsearch.
3. **Kibana:** A visualization layer that works on top of Elasticsearch.

Primary Purpose: The ELK stack is a set of three tools that can work together to help SREs retrieve and visualize logs and data from various sources in real-time.

How it aids the SRE function:

- Centralized Logging: Aggregates logs from various applications and infrastructure, providing a unified view.
- Real-time Analysis: Enables real-time insights into operational health and application behaviors.

Key Feature Demonstration:

- Sending logs with Logstash: Demonstrate how to configure Logstash to ingest logs or data from a source and send it to Elasticsearch.
- Visualizing with Kibana: Show how to set up a basic dashboard in Kibana to visualize the ingested logs/data.

Troubleshooting:

- Execute a search query in Kibana to trace an error or anomaly across multiple logs to find its root cause.

Here are some essential links to get started with the ELK stack:

Elasticsearch:

1. Official Website: https://www.elastic.co/elasticsearch/
2. Documentation: https://www.elastic.co/guide/en/elasticsearch/reference/index.html
3. Getting Started: https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started.html

Logstash:

4. Official Website: https://www.elastic.co/logstash
5. Documentation: https://www.elastic.co/guide/en/logstash/current/index.html
6. Getting Started: https://www.elastic.co/guide/en/logstash/current/getting-started-with-logstash.html

Kibana:

7. Official Website: https://www.elastic.co/kibana
8. Documentation: https://www.elastic.co/guide/en/kibana/current/index.html
9. Getting Started: https://www.elastic.co/guide/en/kibana/current/setup.html

Elastic Stack (ELK) Overview:

10. Official Overview: https://www.elastic.co/what-is/elk-stack

Tutorials & Resources:

11. Elasticsearch, Logstash, Kibana (ELK) Docker image: https://elk-docker.readthedocs.io/
12. Elastic Learning Portal (free courses available): https://www.elastic.co/training/

4. Grafana (Bonus tool) https://grafana.com

Primary Purpose: Grafana is an open-source platform for monitoring and observability, allowing you to visualize and alert on metrics and logs.

How it aids the SRE function:

- Unified Dashboards: Consolidates data from various sources, offering a centralized view.
- Alerting: Can send notifications based on data thresholds.

Key Feature Demonstration:

- Creating a Dashboard: Demonstrate how to create a Grafana dashboard, adding panels, and pulling in data from sources like Prometheus.
- Setting up Alerts: Show how to set thresholds for data in Grafana to trigger alerts.

Troubleshooting:

- Use Grafana to correlate metrics from multiple sources and identify a system bottleneck or performance degradation.

Prepare a report with screenshots of completed tasks

Tips:

- Ensure that all screenshots are clear and readable. It might be beneficial to annotate screenshots to highlight important sections.
- For lengthy processes, consider providing a combination of screenshots and step lists.
- Always double-check sensitive information. Blur or mask sensitive data in screenshots before including them in the report.
- Once you've gathered all the content and screenshots, use a tool like Microsoft Word, Google Docs, or any other word processing software to compile them into a cohesive report.

Links:

1. Google's SRE Book - https://sre.google/books/
2. Docker's Official Documentation - https://docs.docker.com/
3. Kubernetes Documentation - https://kubernetes.io/docs/
4. Prometheus Documentation - https://prometheus.io/docs/
5. Nagios Core Documentation - https://assets.nagios.com/downloads/nagioscore/docs/
6. Alertmanager Documentation - https://prometheus.io/docs/alerting/latest/alertmanager/