

Certainly! Here’s a \*\*cheat sheet\*\* of the main commands used in this project, grouped by purpose, with brief descriptions.

---

## \*\*VM & Environment Setup\*\*

| Command | Description |

|---------|-------------|

| `vagrant up` | Start and provision Vagrant VMs |

| `vagrant ssh <vmname>` | SSH into a specific Vagrant VM |

| `ssh vagrant@<ip>` | SSH into a VM using its IP address |

---

## \*\*Ansible\*\*

| Command | Description |

|---------|-------------|

| `ansible-playbook -i chaos\_lab/inventory.ini chaos\_lab/playbooks/gather\_state.yml` | Run a playbook to gather system state from all VMs |

| `ansible-playbook -i chaos\_lab/inventory.ini chaos\_lab/playbooks/install\_node\_exporter.yml` | Install Node Exporter on all VMs |

| `ansible-playbook -i chaos\_lab/inventory.ini <playbook.yml>` | Run any Ansible playbook on your inventory |

---

## \*\*Node Exporter\*\*

| Command | Description |

|---------|-------------|

| `curl http://localhost:9100/metrics` | Check if Node Exporter is running and serving metrics on a VM |

| `sudo systemctl status node\_exporter` | Check Node Exporter service status |

| `sudo systemctl start node\_exporter` | Start Node Exporter service |

| `ls -l /usr/local/bin/node\_exporter` | Verify Node Exporter binary exists |

---

## \*\*Prometheus\*\*

| Command | Description |

|---------|-------------|

| `wget https://github.com/prometheus/prometheus/releases/download/v2.52.0/prometheus-2.52.0.linux-amd64.tar.gz` | Download Prometheus binary |

| `tar -xzf prometheus-2.52.0.linux-amd64.tar.gz` | Extract Prometheus archive |

| `sudo mv prometheus-2.52.0.linux-amd64 /opt/prometheus` | Move Prometheus to /opt |

| `sudo useradd --no-create-home --shell /bin/false prometheus` | Create Prometheus user |

| `sudo chown -R prometheus:prometheus /opt/prometheus` | Set ownership for Prometheus files |

| `cd /opt/prometheus` | Change to Prometheus directory |

| `sudo -u prometheus ./prometheus --config.file=prometheus.yml` | Start Prometheus with config file |

| `ss -tulnp | grep 9090` | Check if Prometheus is listening on port 9090 |

| `ps aux | grep prometheus` | Check if Prometheus process is running |

---

## \*\*Grafana\*\*

| Command | Description |

|---------|-------------|

| `sudo apt-get install -y apt-transport-https software-properties-common wget` | Install prerequisites |

| `wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -` | Add Grafana GPG key |

| `echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee /etc/apt/sources.list.d/grafana.list` | Add Grafana repo |

| `sudo apt-get update` | Update package lists |

| `sudo apt-get install -y grafana` | Install Grafana |

| `sudo systemctl enable --now grafana-server` | Enable and start Grafana service |

| `sudo systemctl status grafana-server` | Check Grafana service status |

---

## \*\*EFK Stack (Elasticsearch, Fluentd, Kibana)\*\*

| Command | Description |

|---------|-------------|

| `sudo apt-get install -y openjdk-11-jre` | Install Java (required for Elasticsearch) |

| `wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -` | Add Elastic GPG key |

| `echo "deb https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list` | Add Elastic repo |

| `sudo apt-get install -y elasticsearch kibana` | Install Elasticsearch and Kibana |

| `sudo systemctl enable --now elasticsearch` | Enable and start Elasticsearch |

| `sudo systemctl enable --now kibana` | Enable and start Kibana |

| `curl -fsSL https://toolbelt.treasuredata.com/sh/install-ubuntu-jammy-td-agent4.sh | sh` | Install Fluentd (td-agent) |

| `sudo td-agent-gem install fluent-plugin-elasticsearch` | Install Fluentd Elasticsearch plugin |

| `sudo systemctl restart td-agent` | Restart Fluentd after config changes |

---

## \*\*General Linux/Debugging\*\*

| Command | Description |

|---------|-------------|

| `ls -l <path>` | List files and permissions |

| `sudo systemctl status <service>` | Check status of any systemd service |

| `sudo systemctl start <service>` | Start any systemd service |

| `sudo pkill <process>` | Kill a running process |

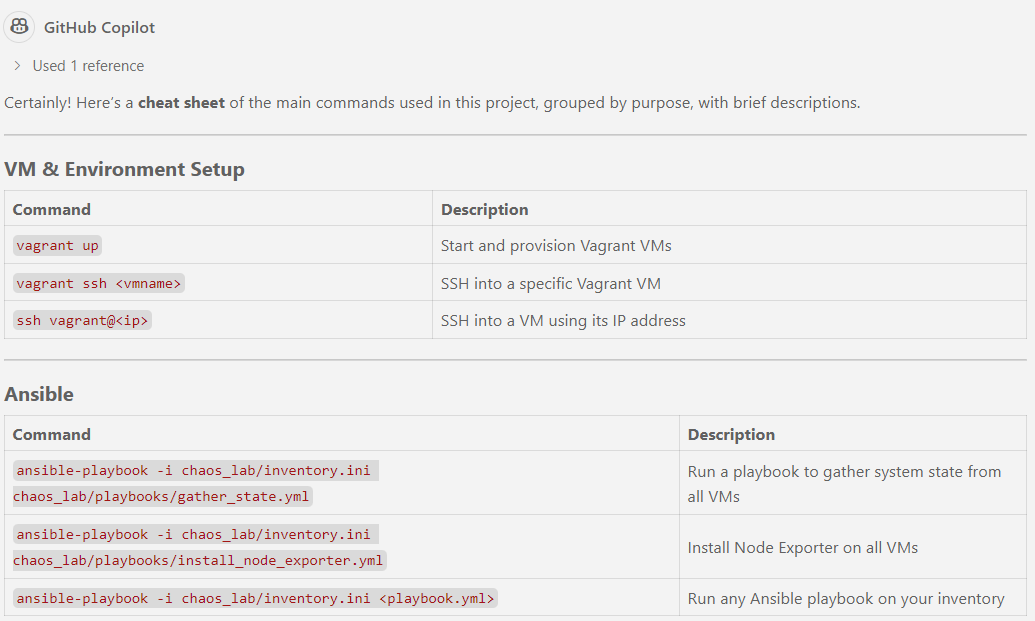
| `sudo ufw allow <port>` | Open a firewall port (if UFW is used) |

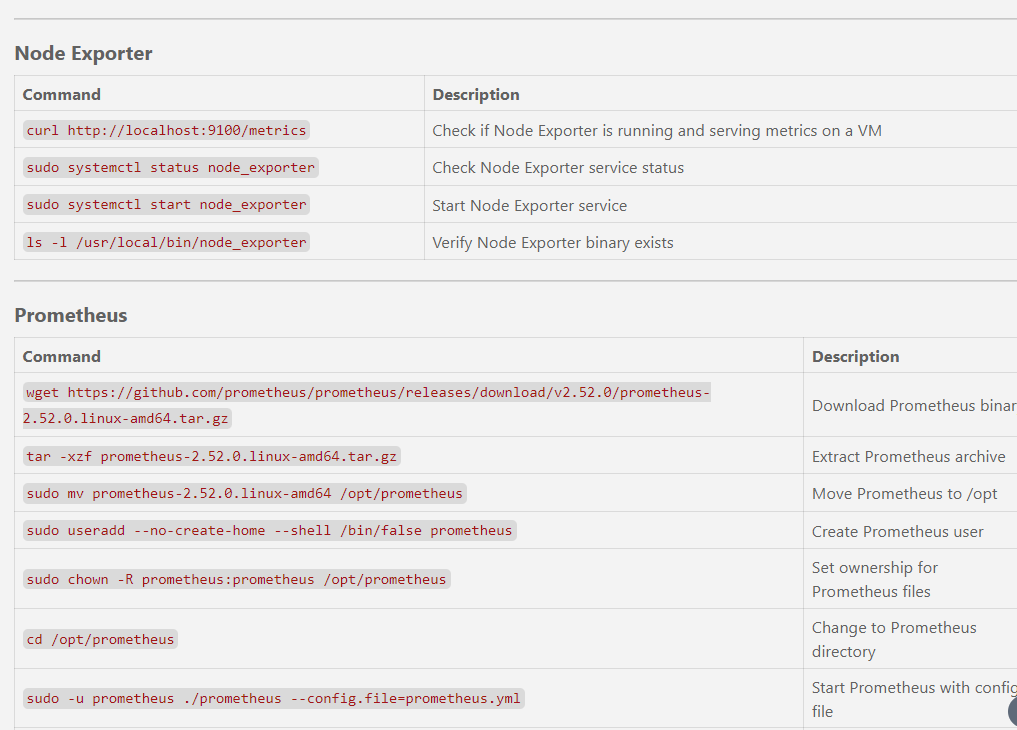
---

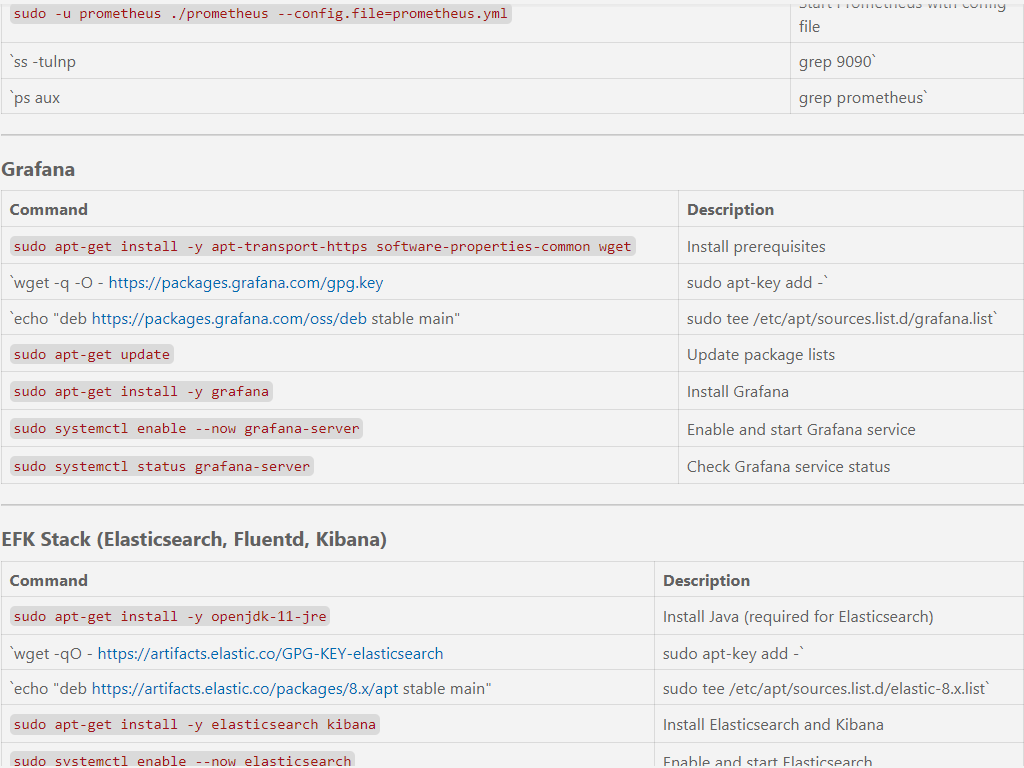
\*\*Tip:\*\*

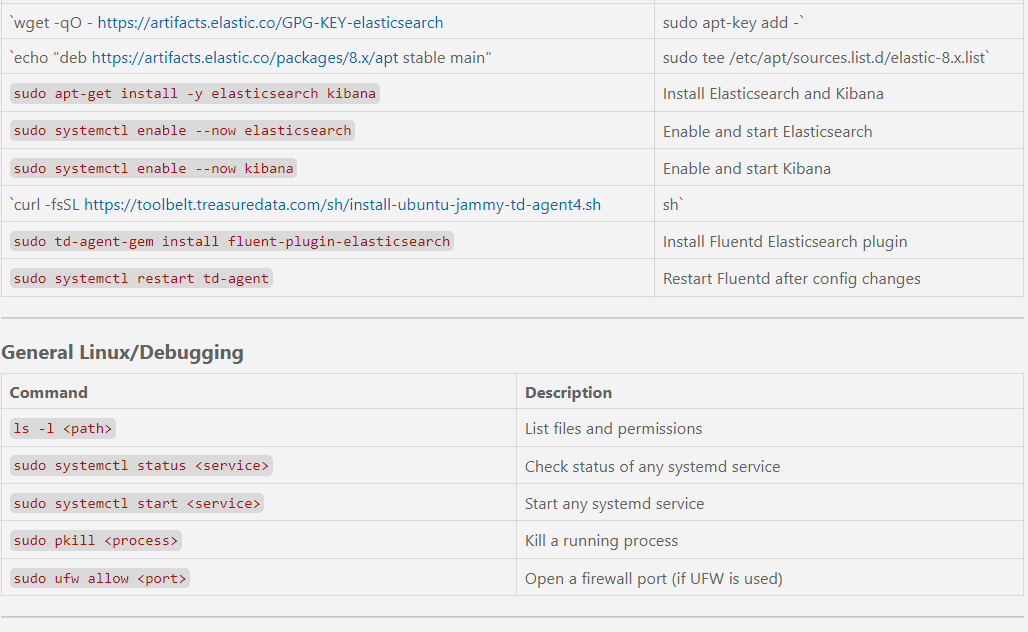
For your presentation, you can copy-paste this table or print it as a quick reference!

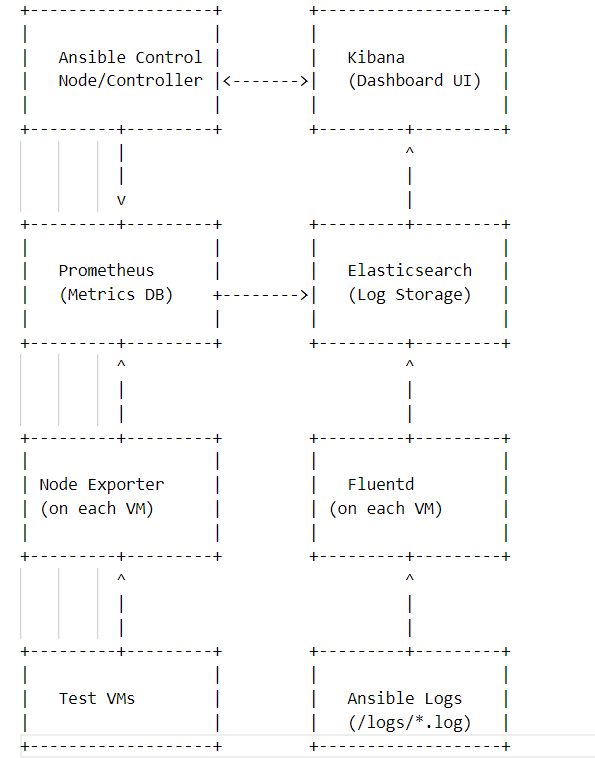
Let me know if you want to add/remove any commands or need more details on any step.











Certainly! Here’s a \*\*workflow diagram\*\* (in text/ASCII and a description) of what you have achieved so far, showing the flow from infrastructure to observability and log aggregation.

---

### \*\*Workflow Diagram (Text/ASCII)\*\*

```

+-------------------+ +-------------------+

| | | |

| Ansible Control | | Kibana |

| Node/Controller |<------->| (Dashboard UI) |

| | | |

+---------+---------+ +---------+---------+

| ^

| |

v |

+---------+---------+ +---------+---------+

| | | |

| Prometheus | | Elasticsearch |

| (Metrics DB) +-------->| (Log Storage) |

| | | |

+---------+---------+ +---------+---------+

^ ^

| |

| |

+---------+---------+ +---------+---------+

| | | |

| Node Exporter | | Fluentd |

| (on each VM) | | (on each VM) |

| | | |

+---------+---------+ +---------+---------+

^ ^

| |

| |

+---------+---------+ +---------+---------+

| | | |

| Test VMs | | Ansible Logs |

| | | (/logs/\*.log) |

+-------------------+ +-------------------+

```

---

### \*\*Workflow Description\*\*

1. \*\*Test VMs\*\*

- Run your applications and services.

- Each VM runs:

- \*\*Node Exporter\*\* (for metrics)

- \*\*Fluentd\*\* (for log shipping)

2. \*\*Node Exporter\*\*

- Collects system metrics (CPU, memory, disk, etc.) from each VM.

- Prometheus scrapes these metrics.

3. \*\*Fluentd\*\*

- Tails Ansible log files (and any other logs you configure) on each VM.

- Ships logs to \*\*Elasticsearch\*\*.

4. \*\*Prometheus\*\*

- Runs on the controller.

- Scrapes metrics from all Node Exporters.

- Stores metrics for querying and visualization.

5. \*\*Elasticsearch\*\*

- Runs on the controller.

- Stores and indexes logs received from Fluentd.

6. \*\*Grafana\*\*

- Connects to Prometheus.

- Visualizes metrics in dashboards.

7. \*\*Kibana\*\*

- Connects to Elasticsearch.

- Visualizes and searches logs in dashboards.

---

### \*\*Summary of Flow\*\*

- \*\*Metrics Flow:\*\*

Test VMs → Node Exporter → Prometheus → Grafana

- \*\*Logs Flow:\*\*

Test VMs (Ansible logs) → Fluentd → Elasticsearch → Kibana

- \*\*Control & Automation:\*\*

Ansible controller orchestrates chaos, recovery, and log collection.

---

If you want a graphical diagram (PNG, SVG, etc.), let me know your preferred tool (draw.io, Lucidchart, etc.) and I can provide a diagram description or exportable file!

Absolutely! Here’s how you can present the \*\*scalability and business value\*\* of this model for a corporate environment, and compare it to a manual approach:

---

## \*\*Replicating This Model in a Corporate Environment\*\*

### \*\*Scalability\*\*

- \*\*Current Lab:\*\* 5 VMs

- \*\*Corporate Scale:\*\* 500–5,000+ servers, containers, or cloud instances

- \*\*Tools Used:\*\* Ansible, Prometheus, Grafana, Node Exporter—all are enterprise-ready and scale horizontally.

---

### \*\*Value Added\*\*

#### \*\*1. Infrastructure Automation\*\*

- \*\*Provisioning, configuration, and updates\*\* can be managed centrally for thousands of nodes.

- \*\*Consistency:\*\* Every server is configured identically, reducing “snowflake” servers and configuration drift.

- \*\*Rapid onboarding:\*\* New servers can be added to the fleet with a single command.

#### \*\*2. Resource Optimization\*\*

- \*\*Automated monitoring\*\* helps identify underutilized or failing resources.

- \*\*Proactive scaling:\*\* Easily automate scaling up/down based on real metrics.

- \*\*Faster incident response:\*\* Automated chaos and recovery reduce downtime.

#### \*\*3. Maintenance & Operations\*\*

- \*\*Centralized playbooks:\*\* One source of truth for all operational tasks.

- \*\*Scheduled tasks:\*\* Regular audits, patching, and health checks can be automated.

- \*\*Self-healing:\*\* Automated recovery playbooks can fix common issues without human intervention.

#### \*\*4. Observability\*\*

- \*\*Unified dashboards:\*\* Real-time visibility across the entire infrastructure.

- \*\*Historical data:\*\* Trends and anomalies are easy to spot, supporting capacity planning and compliance.

---

### \*\*Manual Effort vs. Automated Model\*\*

| Task | Manual (5,000 nodes) | Automated (Ansible/Prometheus) |

|-----------------------------|------------------------------|-------------------------------------|

| Initial setup/config | Weeks/months, error-prone | Hours/days, repeatable |

| Routine audits | Dozens of engineers, slow | Scheduled, 1 engineer can manage |

| Chaos testing | Risky, hard to coordinate | Safe, repeatable, low risk |

| Recovery from failure | Minutes–hours per incident | Seconds–minutes, auto-remediation |

| Monitoring | Fragmented, siloed | Centralized, real-time |

| Documentation | Often outdated/manual | Playbooks are living documentation |

---

### \*\*Business Value\*\*

- \*\*Reduced downtime:\*\* Automated detection and recovery mean less business impact.

- \*\*Lower operational costs:\*\* Fewer engineers needed for routine tasks.

- \*\*Faster innovation:\*\* Teams can focus on new features, not firefighting.

- \*\*Auditability & compliance:\*\* Automated logs and dashboards support regulatory needs.

- \*\*Future-proof:\*\* Easily extend to cloud, containers, or hybrid environments.

---

### \*\*Summary Statement\*\*

> “By automating chaos engineering, monitoring, and recovery at scale, we move from reactive, manual firefighting to proactive, resilient operations. This model not only saves time and reduces errors, but also empowers teams to manage thousands of resources with the same confidence and speed as a small lab environment.”

---

Let me know if you want this in slide format or need more details for your presentation!

Certainly! Here’s a \*\*comprehensive summary\*\* you can use for your presentation and self-study notes. This covers your goals, achievements, challenges, learning outcomes, and a stepwise technical summary.

---

## \*\*Project Summary: Chaos Engineering & Observability Lab\*\*

### \*\*Goal\*\*

- \*\*Build a hands-on lab\*\* to practice chaos engineering, observability, and automated recovery using open-source tools.

- \*\*Demonstrate resilience\*\*: Show how systems behave under failure and how quickly they can recover.

- \*\*Learn and showcase\*\*: Infrastructure automation, monitoring, and troubleshooting using Ansible, Prometheus, and Grafana.

---

### \*\*What We Achieved\*\*

- \*\*Automated VM provisioning\*\* and software installation.

- \*\*Centralized configuration management\*\* with Ansible.

- \*\*Automated chaos injection\*\* (simulate failures).

- \*\*Automated state collection and logging\*\* before, during, and after chaos.

- \*\*Real-time monitoring\*\* with Prometheus and Grafana.

- \*\*Self-healing and recovery\*\* using Ansible playbooks.

- \*\*Stepwise, repeatable workflow\*\* for chaos experiments and audits.

---

### \*\*Stepwise Technical Summary\*\*

#### \*\*1. Environment Setup\*\*

- \*\*Created multiple VMs\*\* (test targets + controller) using Vagrant/VirtualBox.

- \*\*Installed required software\*\*: Python, Ansible, Node Exporter, Prometheus, Grafana.

- \*\*Configured SSH access\*\* from controller to all VMs for Ansible automation.

#### \*\*2. Ansible Automation\*\*

- \*\*Inventory management\*\*: Defined all VMs in Ansible inventory.

- \*\*Playbooks for:\*\*

- Gathering system state (memory, disk, packages).

- Injecting chaos (e.g., stopping services, simulating resource exhaustion).

- Recovery/cleanup (restoring services, cleaning up).

- Fetching logs from VMs to controller.

- \*\*Added timestamps\*\* to logs for precise measurement of task durations.

#### \*\*3. Observability & Monitoring\*\*

- \*\*Installed Node Exporter\*\* on all VMs to expose system metrics.

- \*\*Installed Prometheus\*\* on the controller to scrape metrics from all VMs.

- \*\*Installed Grafana\*\* for visualization.

- \*\*Connected Grafana to Prometheus\*\* and imported dashboards for real-time monitoring.

---

### \*\*What We Are Achieving / Business Value\*\*

- \*\*Proactive resilience testing\*\*: Find weaknesses before real incidents.

- \*\*Automated, repeatable chaos experiments\*\*: No manual steps, less human error.

- \*\*Centralized monitoring and logging\*\*: Faster troubleshooting and root cause analysis.

- \*\*Skill development\*\*: Modern DevOps, SRE, and automation practices.

---

### \*\*Difficulties & Advanced Aspects\*\*

- \*\*Complex Ansible playbooks\*\*: Handling variables, file transfers, error handling.

- \*\*Service orchestration\*\*: Ensuring correct order of chaos, recovery, and monitoring.

- \*\*Debugging distributed systems\*\*: Tracking logs and metrics across multiple VMs.

- \*\*Integrating multiple tools\*\*: Ansible, Prometheus, Grafana, Node Exporter.

- \*\*Learning curve\*\*: Advanced Ansible features (register, with\_items, handlers, etc.), systemd, Linux networking.

---

### \*\*Learning Outcomes\*\*

- \*\*Deep understanding of Ansible\*\*: Playbooks, roles, variables, modules, error handling.

- \*\*Hands-on with monitoring stack\*\*: Prometheus and Grafana.

- \*\*Practical chaos engineering\*\*: Simulating and recovering from failures.

- \*\*End-to-end automation\*\*: From VM creation to monitoring and recovery.

---

### \*\*Complete Picture: End-to-End Flow\*\*

1. \*\*Provision VMs\*\* (Vagrant/VirtualBox).

2. \*\*Install software\*\* (Python, Ansible, Node Exporter, etc.).

3. \*\*Configure SSH\*\* for Ansible access.

4. \*\*Write and run Ansible playbooks\*\* for:

- State gathering

- Chaos injection

- Recovery

- Log collection

5. \*\*Install and configure Prometheus & Grafana\*\* on controller.

6. \*\*Visualize metrics and logs\*\* in Grafana.

7. \*\*Repeat chaos experiments\*\* and measure system resilience.

---

### \*\*How Advanced Is This?\*\*

- \*\*Intermediate to advanced\*\* DevOps/SRE project.

- Combines infrastructure automation, chaos engineering, and observability.

- Real-world skills for production environments.

---

\*\*Tip:\*\*

For your presentation, include screenshots of your dashboards, sample logs, and playbook snippets to show your hands-on work!

---

Let me know if you want a slide outline or more details on any section!