**Jenkins Automated Deployment Pipeline Documentation**

**Overview**

The following documentation details a Jenkins pipeline that automates the deployment of a MicroK8s environment on an AWS EC2 instance. Once the environment is provisioned, SonarQube, a continuous inspection tool, is deployed on the MicroK8s cluster via Helm.

**Prerequisites**

* Jenkins with Pipeline and necessary plugins installed.
* AWS credentials added securely to Jenkins.
* Properly set SSH keys for remote server access.
* Pipeline Workflow

**Workspace Cleanup and Repository Clone:**

**Objective:** Ensure a clean workspace for a fresh deployment and get the latest code/configuration from the repository.

**Steps:**

* Cleanup the Jenkins workspace.
* Clone the GitHub repository: <https://github.com/zinougeh/zino_sec_comp_take_home>.

**Terraform EC2 Provisioning:**

**Objective: Launch an AWS EC2 instance using Terraform.**

**Steps:**

* Initialize the Terraform directory.
* Apply the Terraform configuration to create an EC2 instance.
* Extract the public IP of the newly created EC2 instance and set it as an environment variable.

**MicroK8s Installation using Ansible:**

**Objective:** Automate the installation of MicroK8s on the EC2 instance using Ansible.

**Steps:**

* Generate a temporary Ansible inventory file containing the EC2 instance's IP address.
* Disable SSH host key checking (Note: Be cautious; this can expose you to man-in-the-middle attacks).
* Execute the Ansible playbook to install MicroK8s.

**SonarQube Deployment via Helm:**

**Objective:** Deploy SonarQube on the MicroK8s cluster using Helm.

**Steps:**

* Copy the SonarQube Helm charts to the EC2 instance.
* Use Helm to install SonarQube on the MicroK8s cluster.
* Verify the Helm installation by listing deployed Helm releases.

**Best Practices and Security**

**Credentials**: Avoid hardcoding credentials in scripts. Instead, use Jenkins credentials binding to inject them securely.

**SSH Host Key Checking**: Disabling SSH host key checking can expose you to potential security risks. Always understand the implications before disabling it.

**Regular Audits**: Periodically review and audit the pipeline for any changes and ensure that best practices are followed.

**Monitoring and Alerts**: Implement monitoring and alerts for the EC2 instance and SonarQube application to ensure uptime and capture any anomalies.

**Maintenance and Troubleshooting**

**Logs:** Jenkins provides comprehensive logs for each pipeline run. In case of failures, the logs can provide vital information to diagnose issues.

**Updates**: Periodically check for updates to tools and plugins, such as Terraform, Ansible, Helm, etc. Keeping tools up to date ensures security patches are applied and benefits from new features.

Incorporating these guidelines and following this workflow should provide a seamless and automated deployment experience. Always ensure that you conduct a thorough review and test the pipeline in a non-production environment before deploying to a live environment.