**Design Document: Automated Change Request (CR) Creation and Approval with ServiceNow and Terraform**

**1. Introduction**

* **1.1. Purpose:**
  + This document outlines the design and implementation of an automated Change Request (CR) creation and approval system using Terraform and ServiceNow.
  + It aims to streamline the deployment process by automating the CR lifecycle, reducing manual effort, and improving auditability.
* **1.2. Scope:**
  + This document covers the design of the Terraform modules, the Jenkins pipeline, and the GitHub Actions workflow involved in the automated CR process.
  + It also describes the integrations with ServiceNow and the necessary configurations.
* **1.3. Audience:**
  + This document is intended for project stakeholders, developers, operations teams, and IT management involved in the deployment and change management processes.

**2. Business Requirements and Use Case**

* **2.1. Problem Statement:**
  + Manual CR creation and approval processes are time-consuming and prone to errors.
  + Lack of automation leads to delays in deployments and increases operational overhead.
  + The need for a streamlined, auditable, and repeatable process for managing infrastructure changes.
* **2.2. Use Case:**
  + When a pull request (PR) is opened in the GitHub repository containing Terraform code, a CR is automatically created in ServiceNow.
  + Jenkins pipeline is triggered to execute Terraform plan/apply/destroy actions, and also monitor the change request status.
  + Once the CR is approved in ServiceNow, the Jenkins pipeline proceeds with the deployment.
  + Email notifications are sent to relevant stakeholders at each stage of the process, including CR creation and approval.
* **2.3. Benefits:**
  + Reduced manual effort and faster deployment cycles.
  + Improved auditability and compliance with change management policies.
  + Enhanced collaboration and communication between development and operations teams.
  + Improved consistency and reduced human error.

**3. Technical Design**

* **3.1. Architecture Overview:**
  + The solution comprises the following components:
    - GitHub: Source code repository and PR triggering.
    - Terraform: Infrastructure as Code (IaC) tool for managing resources and creating CRs.
    - ServiceNow: IT service management platform for CR tracking and approval.
    - Jenkins: Automation server for pipeline execution and integration.
    - Github actions: Used to create the CR on pull request creation.
* **3.2. Terraform Modules:**
  + **modules/servicenow\_cr/:**
    - **main.tf:** Uses the http provider to create a CR in ServiceNow via the REST API.
    - **variables.tf:** Defines input variables such as ServiceNow instance URL, credentials, and CR details.
    - **outputs.tf:** Outputs the CR ID created in ServiceNow.
  + **main.tf (Root Module):**
    - Calls the servicenow\_cr module and defines global variables.
    - Outputs the created CR ID.
  + **providers.tf:** Configures the http provider.
  + **terraform.tfvars:** Stores variable values for the Terraform deployment.
* **3.3. Jenkins Pipeline:**
  + **Jenkinsfile:**
    - Defines the pipeline stages for Terraform execution and CR approval.
    - Uses withCredentials to securely inject ServiceNow and GCP credentials.
    - Uses the http provider to create the CR.
    - Retrieves the CR ID from Terraform output.
    - Checks the CR approval status in ServiceNow via the REST API.
    - Sends email notifications using the Email Extension Plugin.
    - Uses terraform workspaces to manage different enviornments.
  + **Pipeline Stages:**
    - **Check Repo Subdirectories:** Verifies the existence of necessary directories.
    - **Setup & Terraform Execution:** Executes Terraform commands (init, plan, apply, destroy).
    - **Debug Info:** Provides debugging information.
    - **Check CR Approval & Notify:** Checks CR status and sends notifications.
* **3.4. GitHub Actions:**
  + **.github/workflows/cr.yml:**
    - Triggers on PR creation.
    - Sets up Terraform and applies the Terraform code to create the CR.
    - Uses GitHub secrets to store ServiceNow credentials.

**4. Implementation Details**

* **4.1. ServiceNow Integration:**
  + Uses the ServiceNow REST API to create and check the status of CRs.
  + Requires a ServiceNow user with appropriate permissions.
  + Credentials are stored securely in Jenkins and GitHub Secrets.
* **4.2. Terraform Configuration:**
  + Uses the http provider to interact with the ServiceNow API.
  + Variables are defined in variables.tf and values are stored in terraform.tfvars.
* **4.3. Jenkins Configuration:**
  + Requires the Email Extension Plugin and relevant credentials configured.
  + Uses the withCredentials step for secure credential injection.
  + Terraform is installed on the Jenkins agent.
* **4.4. GitHub Actions Configuration:**
  + Uses the hashicorp/setup-terraform action.
  + GitHub secrets are configured with the ServiceNow username and password.

**5. Security Considerations**

* **5.1. Credential Management:**
  + ServiceNow credentials are stored securely in Jenkins and GitHub Secrets.
  + GCP service account keys are managed using Jenkins credentials.
* **5.2. Access Control:**
  + ServiceNow user permissions are configured to limit access to CR creation and status checks.
  + Jenkins pipeline access is controlled based on user roles.
  + Github secrets are only accessible to the github action.
* **5.3. Network Security:**
  + Communication with ServiceNow is secured via HTTPS.
  + Restrict network access to the Jenkins server.

**6. Deployment and Operations**

* **6.1. Deployment Process:**
  + Terraform code is committed to the GitHub repository.
  + A pull request triggers the GitHub Actions workflow to create the CR.
  + The Jenkins pipeline is triggered to execute Terraform and monitor CR status.
  + Email notifications are sent at each stage.
* **6.2. Monitoring and Maintenance:**
  + Jenkins logs are monitored for pipeline execution and errors.
  + ServiceNow CR logs are reviewed for audit purposes.
  + Regular updates are applied to Terraform, Jenkins, and ServiceNow components.

**7. Conclusion**

* This automated CR creation and approval system streamlines the deployment process and improves change management.
* By integrating Terraform, Jenkins, and ServiceNow, the solution reduces manual effort, enhances auditability, and accelerates deployment cycles.
* By using github actions, we are able to create the CR as soon as the pull request is created.

**"Please reach out to me if you have any doubts regarding this document."**