SCALING TERRAFORM DEPLOYMENTS WITH GITHUB ACTIONS: ESSENTIAL CONFIGURATIONS TO GET STARTED

#90DAYSOFDEVOPS

GitHub Repository Recommended Settings

- Enabling a branch protection policy ensures code quality, security, and collaboration.
- We'll discuss the recommended settings, focusing on requiring a pull request before merging.

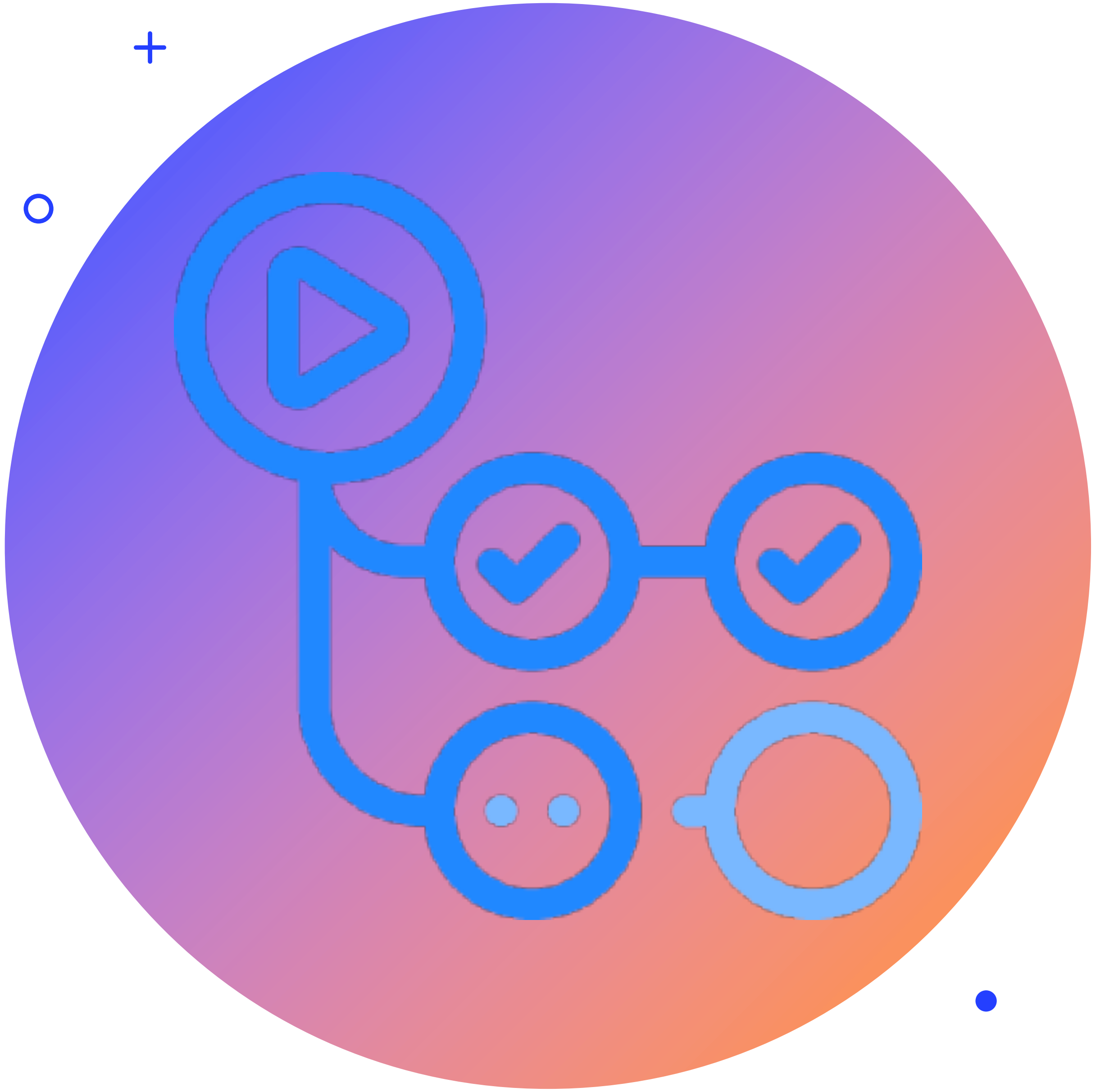
Structuring Your GitHub Repository

- Creating a well-organized folder structure is crucial for maintaining a clean and manageable codebase.
- We'll explore a recommended folder structure for your Terraform GitHub repository, which includes directories for workflows, environments, modules, and platform components.

GitHub Repository Environment Secrets and Variables

- Learn how to manage secrets and variables efficiently for deploying between environments.
- Explore the use of GitHub Repository Environment Secrets and Variables for enhanced security and streamlined deployments.

GitHub Action Setup



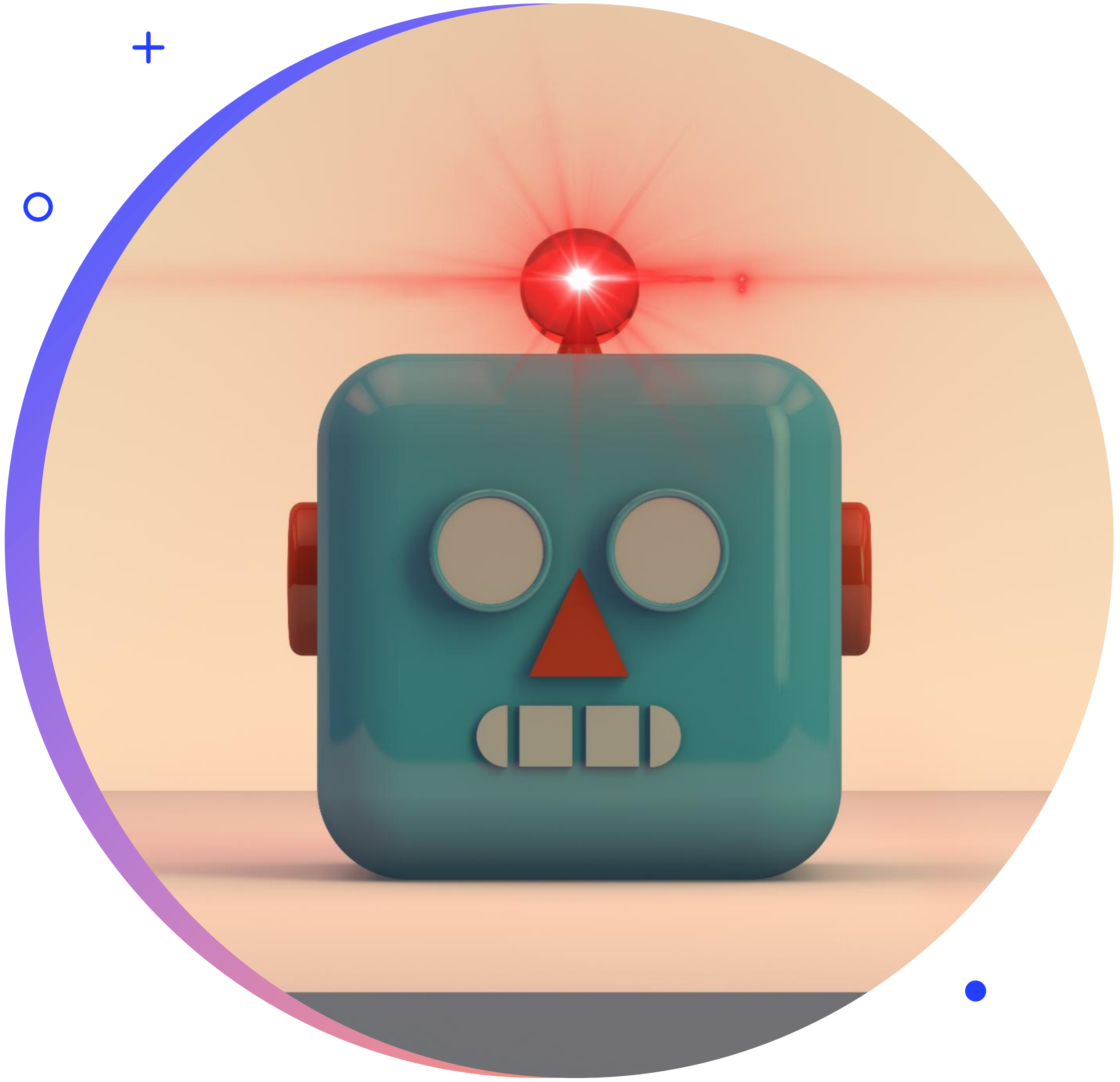
- A deep dive into the core GitHub Action file (deploy.yaml).
- Explanation of key sections, including trigger conditions, matrix strategy, and how to use GitHub repository secrets in the workflow.

GitHub Action Templating

- Understand the benefits of GitHub Action Templating in creating reusable workflows.
- Explore a template file (template-terraform.yaml) that simplifies the deployment process and encourages consistency.

Using Matrixes to Deploy Multiple Components to Multiple Environments within GitHub Actions

- GitHub Actions matrixes are a powerful tool for managing multi-environment deployments efficiently.
- Learn how to customize Terraform deployments for different component-environment combinations using matrix variables.



Dependabot Setup

- Ensure consistency across your Terraform codebase by automating dependency updates with Dependabot.
- Integrate Dependabot into GitHub Actions and configure it for scheduled updates.