TOOL EXPLORATION REPORT DRONE CI

Muhammad Ahmad 20F-0125

Report on Drone CI Tool Exploration

1. Introduction to Drone CI

Drone CI is an open-source CI/CD platform designed to automate the software development and deployment process. It allows developers to define build pipelines, run tests, and deploy applications efficiently. Drone CI is known for its flexibility, extensibility, and support for containerization technologies.

2. Purpose and Importance of CI/CD

CI/CD is essential in modern software development to ensure code quality, shorten development cycles, and deliver reliable software to end-users. Drone CI plays a critical role in achieving these objectives by automating repetitive tasks and providing a streamlined process for code integration and delivery.

3. Features of Drone CI

Build Pipelines

Drone CI enables the creation of complex build pipelines that include various stages such as building, testing, and deployment. This allows for thorough automation of the software delivery process.

Scalability

Drone CI can be easily scaled to accommodate projects of varying sizes and complexities. Its architecture supports the distribution of workloads across multiple agents and runners.

Containerization

Drone CI leverages containerization technologies like Docker to ensure consistent and reproducible builds across different environments.

Multi-Platform Support

Drone CI supports a wide range of operating systems and platforms, making it suitable for diverse project requirements.

Extensive Plugin Ecosystem

The tool boasts an extensive plugin ecosystem, allowing users to extend its functionality easily. These plugins cover integrations with various services, notification mechanisms, and deployment targets.

4. Installation and Configuration

Prerequisites

Installation prerequisites include an Ubuntu server, Docker, Docker Compose, and Git. Additional prerequisites depend on specific project requirements.

Installation Steps

The installation process involves cloning the Drone CI repository, configuring environment variables, and using Docker Compose to deploy the application.

Integration with Version Control Systems (e.g., GitHub)

Drone CI seamlessly integrates with popular version control systems like GitHub, enabling automatic triggering of builds upon code changes.

5. Usage of Drone CI

Creating a `.drone.yml` Configuration File

Users define build and deployment steps in a `.drone.yml` configuration file within their project repositories. This file specifies how Drone CI should handle builds and workflows.

Running Builds and Workflows

Drone CI monitors version control repositories and automatically triggers builds when code changes occur. Users can also manually initiate builds and workflows as needed.

Monitoring and Logging

Drone CI provides monitoring and logging capabilities, offering insights into build and deployment processes. Users can identify issues and track progress through the web interface or logs.

Notifications and Alerts

The tool supports various notification mechanisms, including email, chat platforms (e.g., Slack), and web hook integrations, ensuring teams are informed about build and deployment status.

The Configuration and commands are:

```
docker run \
  --rm \
  --volume=$PWD/data:/data \
  --env=DRONE_GITHUB_CLIENT_ID=e985d46cd8a36b1f0016 \
  --env=DRONE_GITHUB_CLIENT_SECRET=8343324dca241bbc7e3293ae3c83be85e768e32c \
  --env=DRONE_RPC_SECRET=68bd980d21e61ffd3dd32a98b6e83cff \
  --env=DRONE_SERVER_HOST=db0e-223-123-11-106.ngrok-free.app \
  --env=DRONE_SERVER_PROTO=https \
  --publish=8080:80 \
  --detach=true \
  --name=drone \
  --network=drone-net \
  drone/drone:2
```

```
docker run \
--rm \
--volume=/var/run/docker.sock:/var/run/docker.sock \
--env=DRONE_RPC_PROTO=http \
--env=DRONE_RPC_HOST=drone \
--env=DRONE_RPC_SECRET=68bd980d21e61ffd3dd32a98b6e83cff \
--env=DRONE_RUNNER_NAME=my-first-runner \
--publish=3000:3000 \
--detach=true \
--name=runner \
--network=drone-net \
drone/drone-runner-docker:1
```

Drone.yml file:

```yaml
kind: pipeline
name: default

steps:
  - name: install_dependencies
    image: node:14  # Use a Node.js version compatible with your Next.js project
    commands:
      - npm install

  - name: build
    image: node:14
    commands:
      - npx create-next-app my-next-app  # Create a new Next.js app if not already created
      - cd my-next-app
      - npm install  # Install project dependencies

  - name: test
    image: node:14
    commands:
      - cd my-next-app
      - npm test
```