Manage preview environments with a Slack bot

ByteBenders

Who are ByteBenders?

Volodymyr Larkin

Lead

Andrii Petrenko

Implementer

Vitalii Bielykh

Ideas generator

Anton Davitadze

Intern

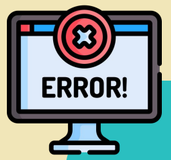
What do you need?

To build a Slack bot for the automated management of preview environments.

What's the problem, what are we solving?

Time spent on manual env. management

Manually creating, configuring, and deleting preview environments requires significant developer time, distracting them from core tasks and slowing the development cycle.



Risk of errors in manual operations

Manual actions often lead to errors that can cause stability and compatibility issues in the environments, making it difficult to test and release new application versions.

Lack of centralized management and monitoring

The lack of a centralized tool for managing and monitoring environments makes it difficult to control their status and availability, which can lead to unforeseen problems during development and testing.

What functionality is required?

Create Environment

Allows users to create a new preview environment for a specified version of the application, including necessary accompanying services.

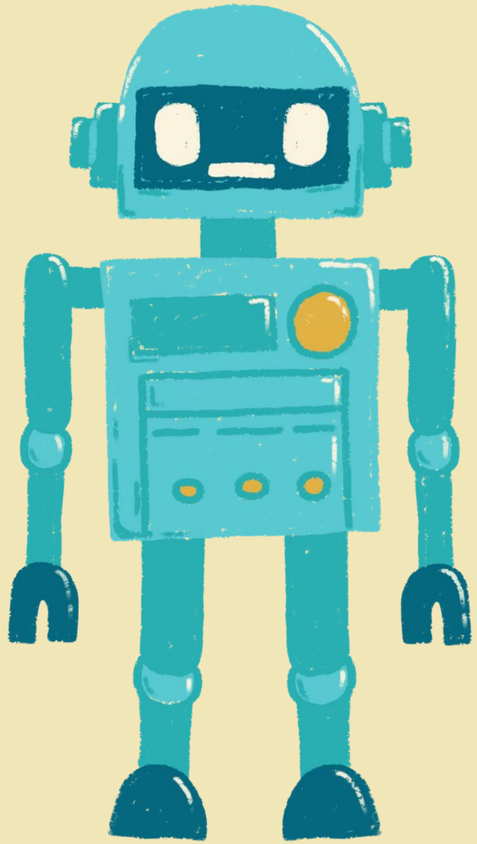
Get Environment Status

Enables users to retrieve a list of environments, the versions of the deployed applications, and their status.

Delete Environment

Allows users to safely delete an environment after demonstration.

Study Planning Immersion



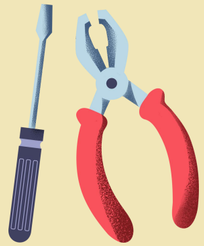
What ready-made solutions are available? How is it implemented?

uffizzi

Simple deployment and management across many environments

bunnyshell

Platform focused on comprehensive tools for provisioning, monitoring and securing cloud resources

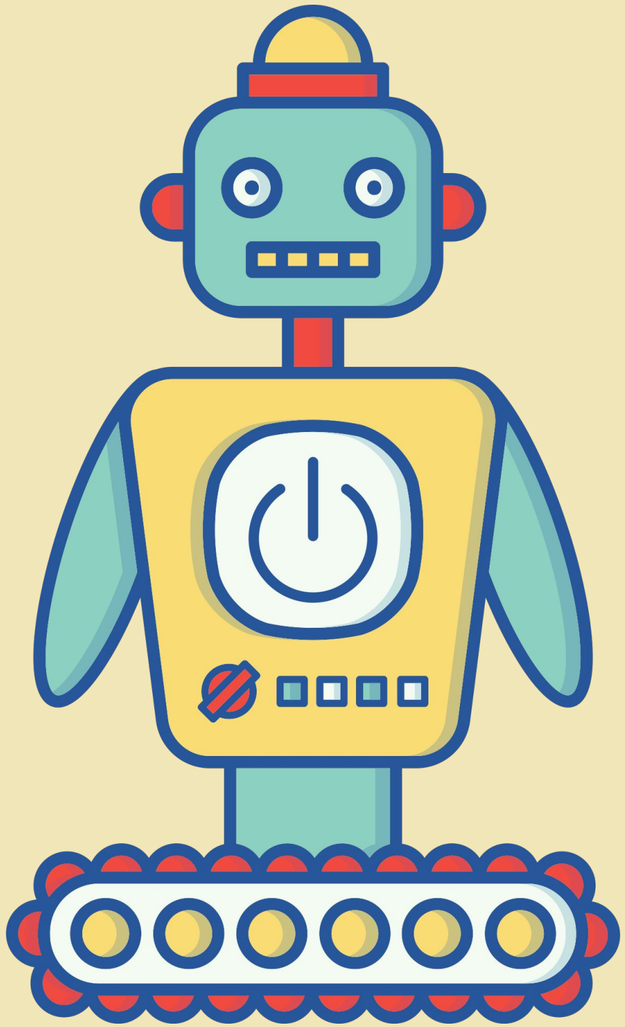


EnvBot

Let's take the best and make our own

+

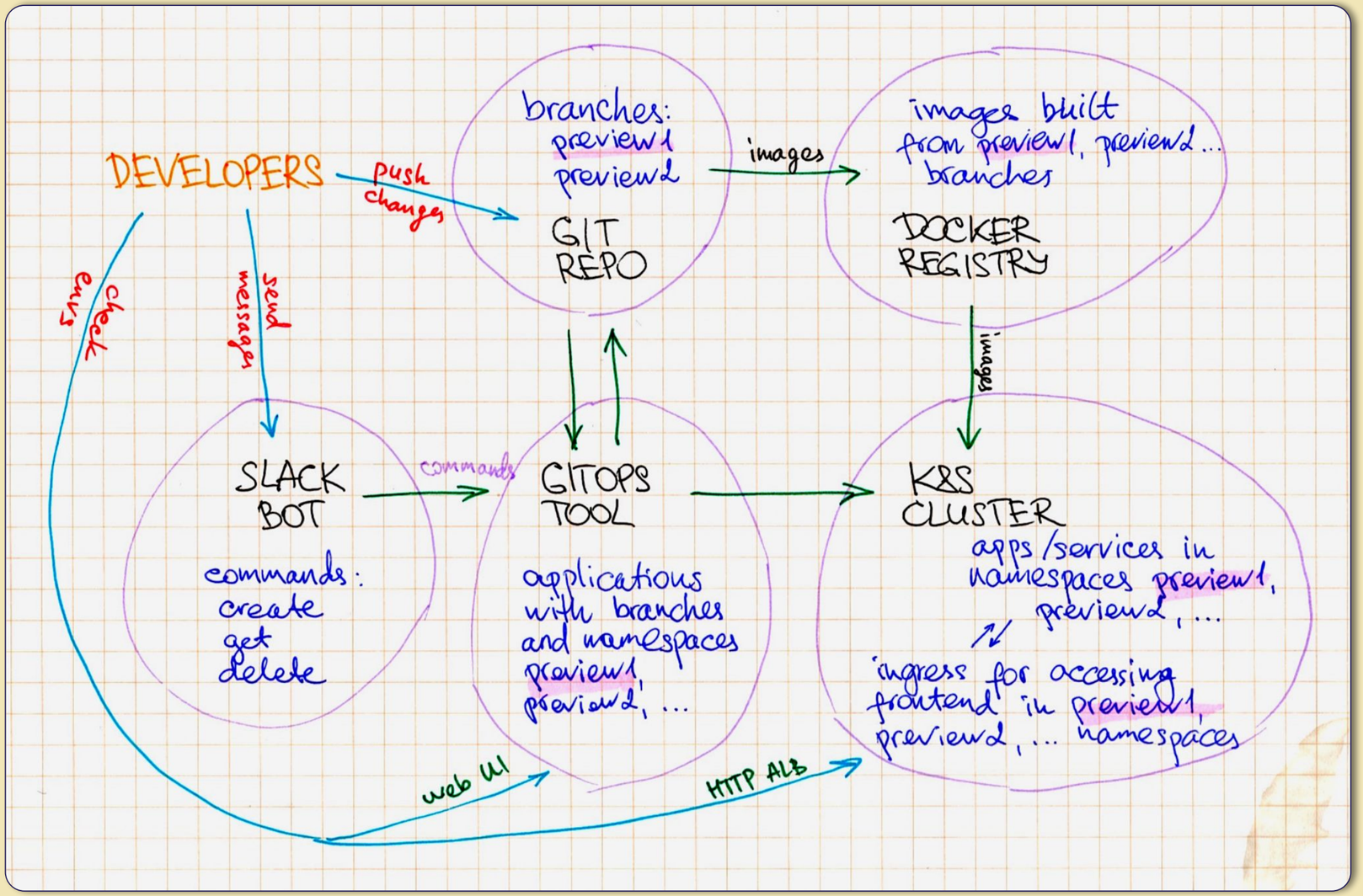
Design



Architectural Decision Records

GitHub
platform for source code, docker images and pipelines

Slack Machine
powerful slack bot framework

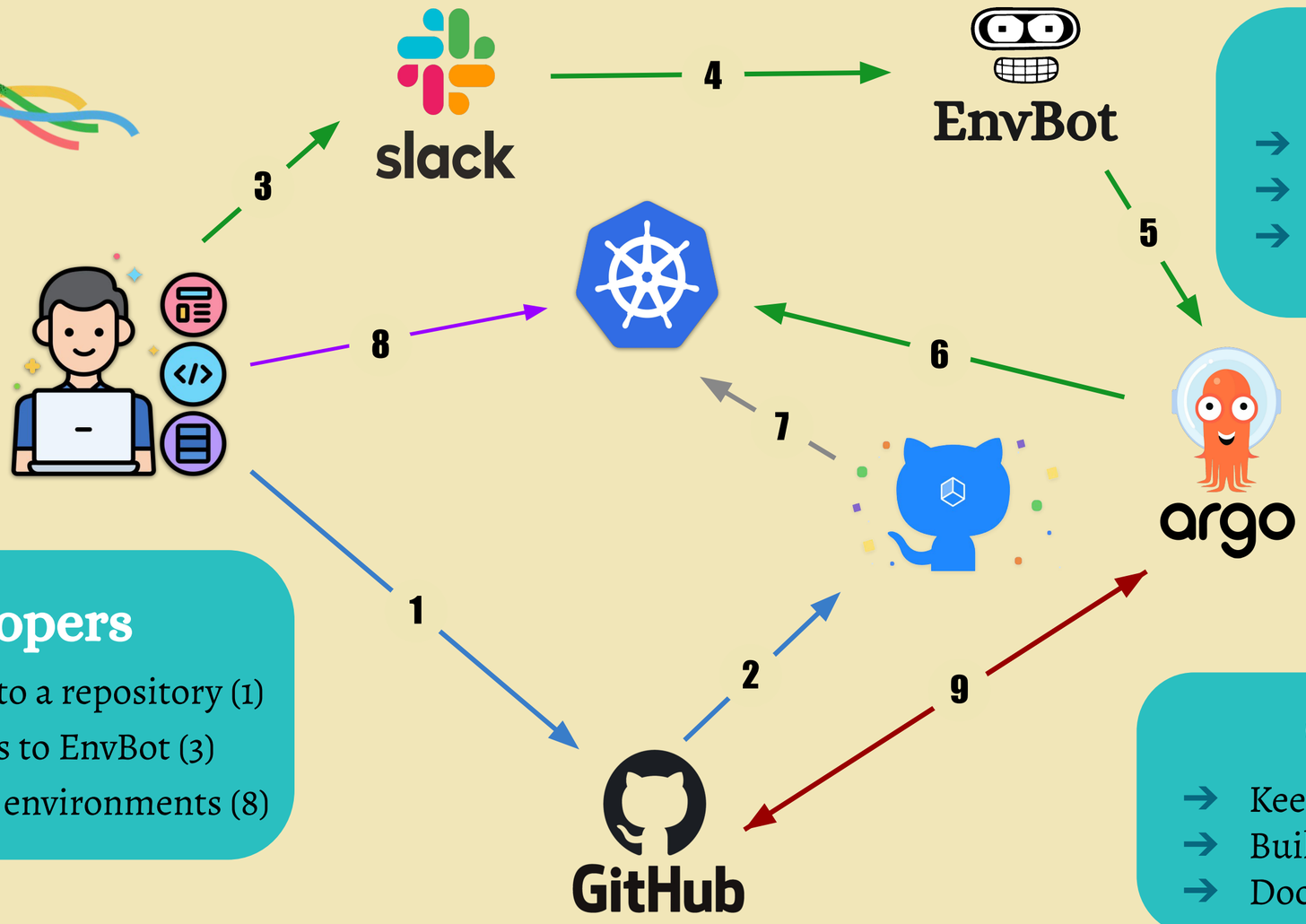


Argo CD
flexible GitOps tool with a nice web UI

GCE VM instance
for runtime

GKE
managed Kubernetes cluster for envs

High-Level Solution Design



ArgoCD

- Accept commands from EnvBot (5)
- Manage environments (6)
- Track changes (9)

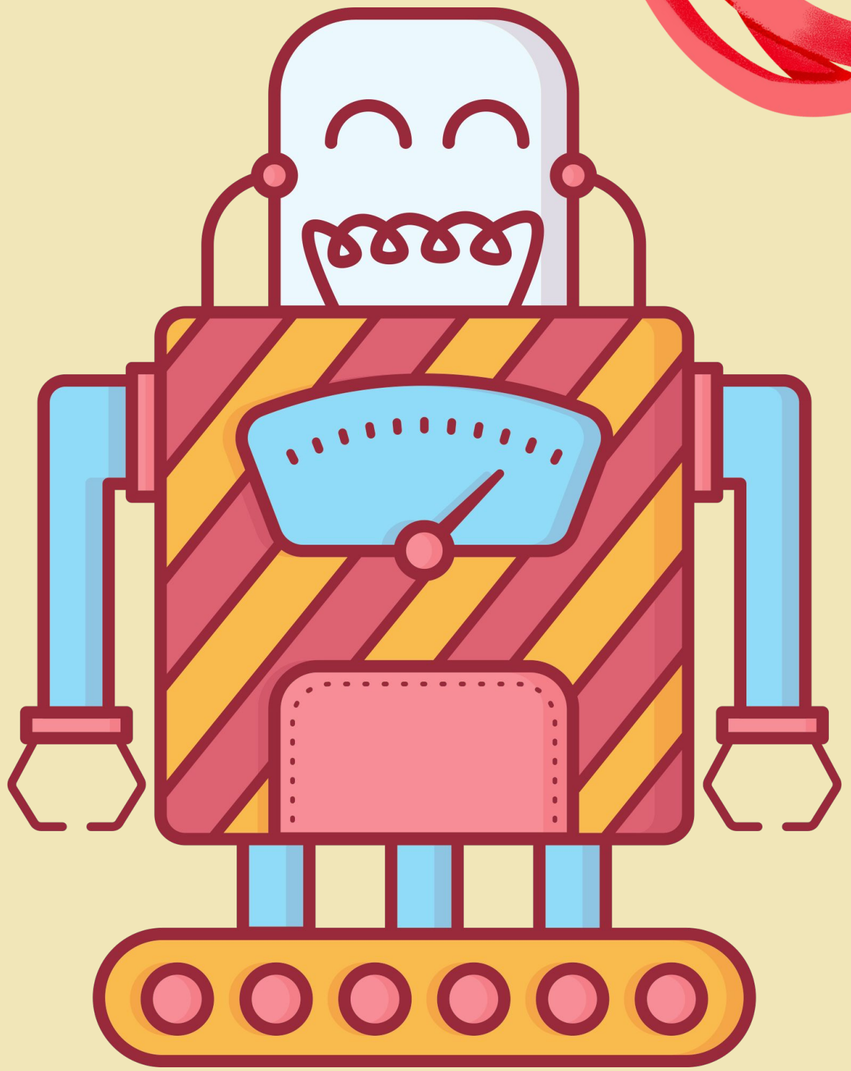
Developers

- Push changes to a repository (1)
- Send messages to EnvBot (3)
- Check created environments (8)

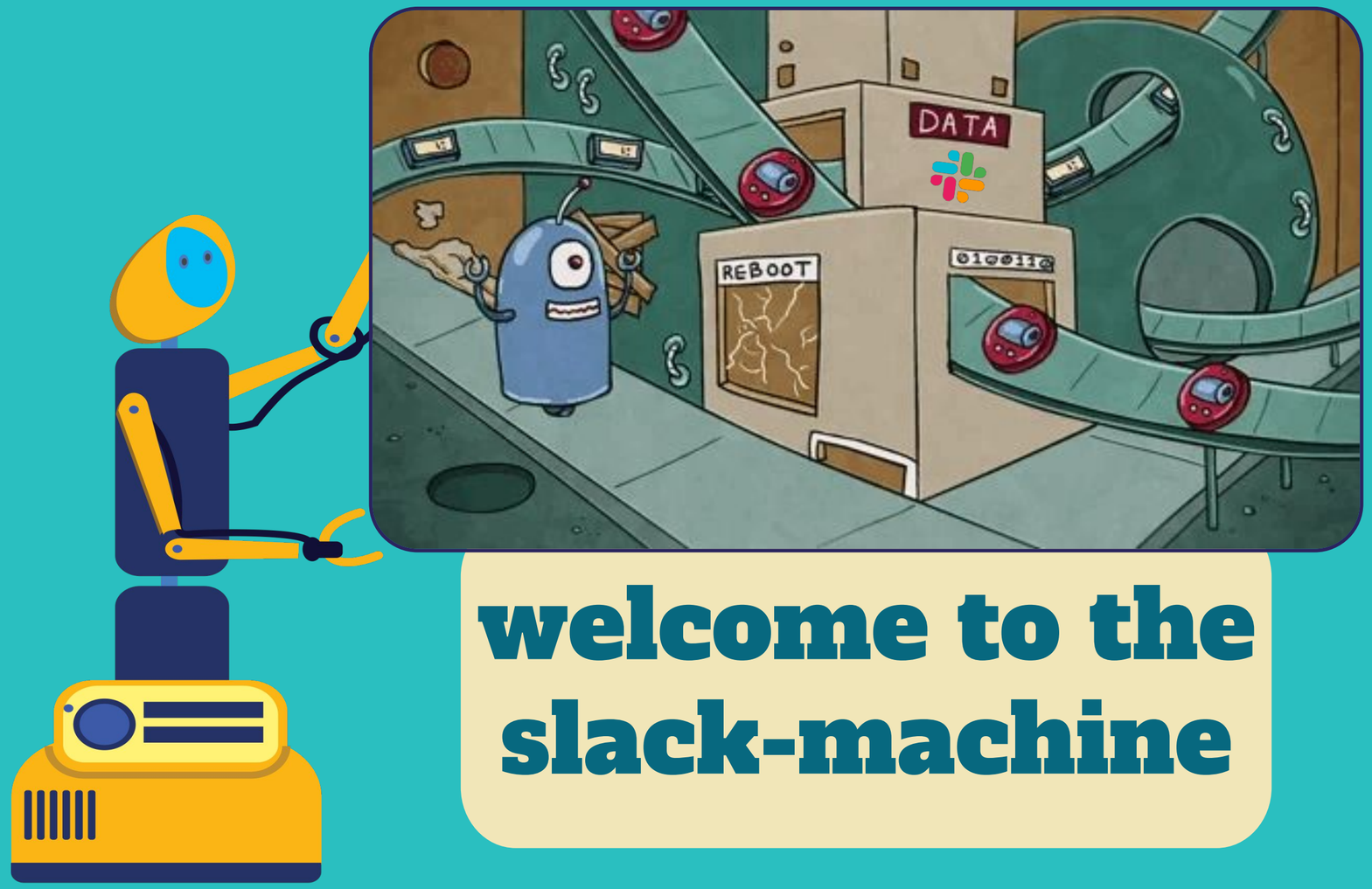
GitHub

- Keep source code
- Build docker images (2)
- Docker registry (7)

Realization



Implementation



welcome to the slack-machine

- Slack Machine is a **simple**, yet **powerful** and **extendable** Slack bot framework.
- That helps you develop your Slack workspace into a **ChatOps powerhouse**.
- Built with an intuitive **plugin system** that allows you to build bots **quickly**, while also enabling **easy** code organization.

How it works?

Step 0: Installation

Slack Machine is published to the Python package index, so you can easily install Slack Machine using pip:

```
pip install slack-machine
```

Step 1: Creating the plugin

In the root directory of your bot (where your **`local_settings.py`** resides), create a plugins folder. Inside it, create an **`__init__.py`** file (to make your plugins folder a package) and a file named **`previewenv.py`**

Your folder structure should now resemble this:

```
├── local_settings.py
└── plugins
    ├── __init__.py
    └── previewenv.py
```

How it works?

Step 2: Adding the code

First, we should import **`MachineBasePlugin`** into our code, along with the **`decorator`** to listen for specific messages. Then, we can create our plugin class that includes a function to listen for The Answer and responds to it:

```
from machine.plugins.base import MachineBasePlugin
from machine.plugins.decorators import listen_to

class CommandsPlugin(MachineBasePlugin):

    @command("/create")
    async def create(self, command):
        """Handle the /create command to create a new application."""
        try:
            args = command.text.split()
            if len(args) != 2:
                await command.say("Usage: /create app_name branch_name_or_commit_hash")
                return
            app_name, revision = args
            app_result = create_argocd_application(app_name,
                                                   command.sender.team_id, revision)
            await command.say(f"Create Preview Environment: {app_result}")
        except Exception as e:
            await command.say(str(e))
```

How it works?

Step 3: Enabling our plugin

Now we can enable our plugin in the configuration file. Your `local_settings.py` should look like this:

```
SLACK_APP_TOKEN = "xapp-my-app-token"
SLACK_BOT_TOKEN = "xoxb-my-bot-token"

PLUGINS = [
    "plugins.previewenv.CommandsPlugin"
]
```

Step 4: Run

To run your bot with the new plugin:

```
$ slack-machine
```

That's all there is to it!

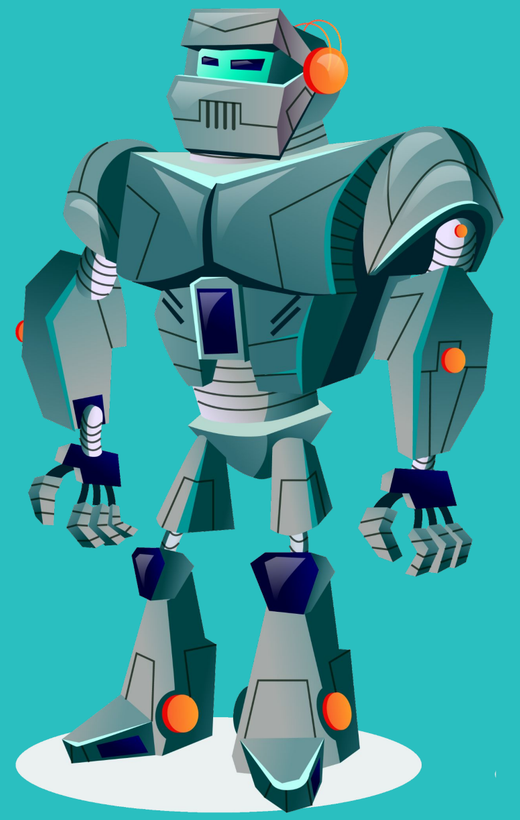
As Slack-Machine is fully built on top of Python's excellent `AsyncIO` library, all the functionality that Slack Machine offers (e.g. sending messages, adding reactions, etc.) is implemented as async functions, a.k.a. *coroutines*. Therefore, the plugin functions you define must also be coroutines.

This will take a while to get used to, but the **reward** is that Slack Machine is able to run your plugin functions **concurrently**.

Upcoming features

- ✓ Creating environments using git branch names or commit hashes
- ❑ Possibility of integration into CI/CD pipelines
- ❑ Using different repositories for deploying into environments
- ❑ Utilizing separate Kubernetes clusters (AWS, GKE, Azure, self-hosted) for deploying resource-demanding preview environments

Before

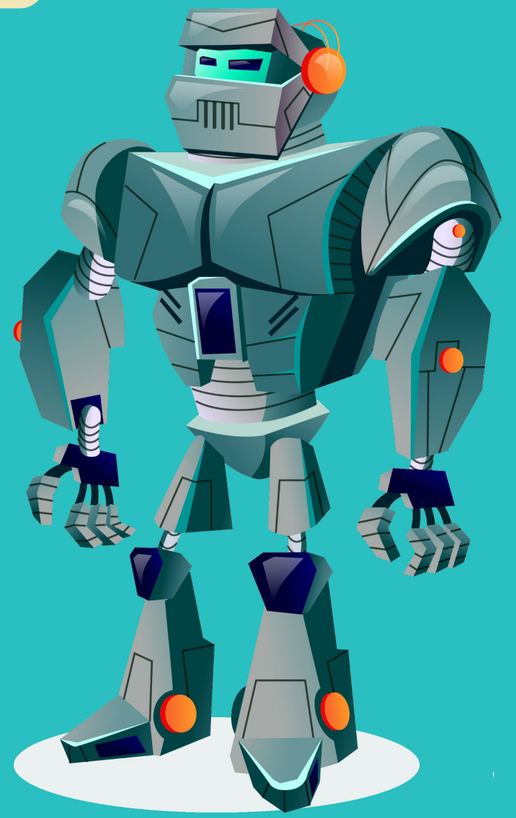


After

Enhanced Architectural Design

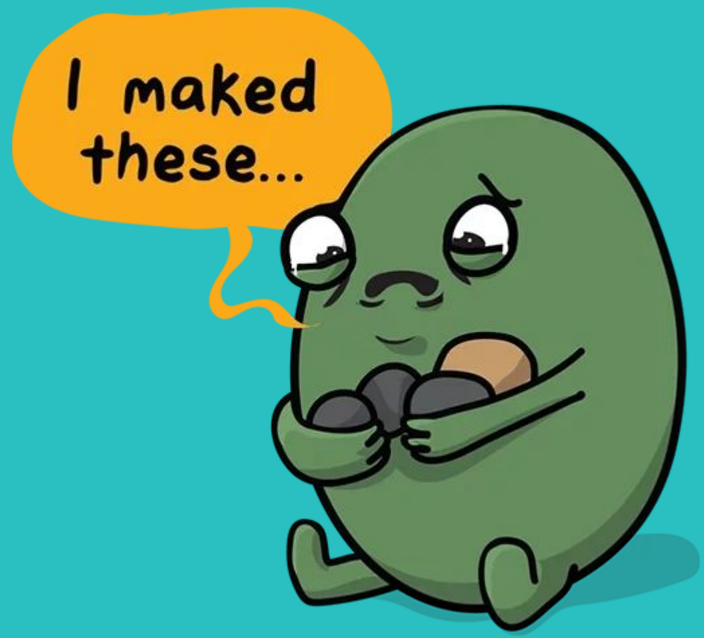
Before

- ❑ Preview environments clean-up
- ❑ Auto-update DNS A records for ALB
- ❑ Extended monitoring and logging
- ❑ Improved security
- ❑ Containerization with using Cloud Run
- ❑ Automated scaling
- ❑ Costs tracking and alerting



After

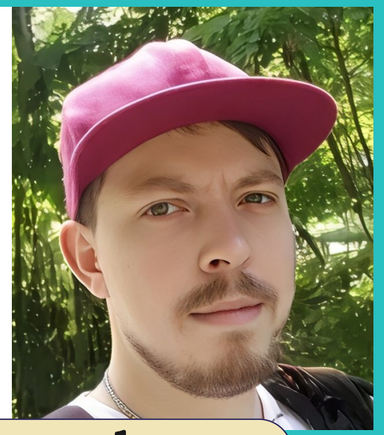
Thank you for your attention!



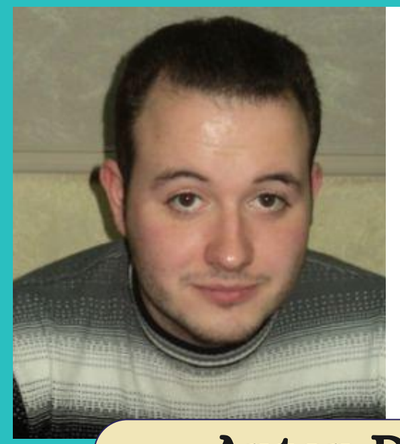
Any questions?

Volodymyr Larkin
vlarkin@gmail.com
Lead

Vitalii Bielykh
Ideas generator



Andrii Petrenko
andrew.petrenko28@gmail.com
Implementer



Anton Davitadze
adavitadze@gmail.com
Intern