Jupyter for DevOps

Moshe Zadka – https://cobordism.com

Acknowledgement of Country

Belmont (in San Francisco Bay Area Peninsula)

Ancestral homeland of the Ramaytush Ohlone people

I live in Belmont, in the San Francisco Bay Area Peninsula. I wish to acknowledge it as the ancestral homeland of the Ramaytush Ohlone people.

Jupyter for DevOps??

Yes!

Data scientists:

Explore...Automate...Share!

DevOps engineers:

Explore...Automate...Share!

Jupyter for DevOps? Isn't Jupyter something for data scientists?

It is! Data scientists need to explore the data. Once they have a sense of the data, they need to automate the data processing. Finally, they need to share their work with their colleagues.

In contrast, DevOps engineers need to explore the system. Once they have a sense of the system, they need to automate their interactions. Finally, they need to share their work with their colleagues.

Similar problems, similar tools. Jupyter has had a long time to mature as a development console, and is well-funded and developed. There is no need for "something like Jupyter, but for DevOps". Jupyter is already that.

So why not jump in and show some examples?

0.1 SSH

SSH

The cause of,and the solution to,all DevOps problems.

SSH is still the way many of us do ad-hoc management of systems. It is a solid server-side general agent platform, and even many "replacements" are built on top of it.

SSHing directly from the command-line is fine. But in modern environments, we can have several, sometimes tens of systems, that we need to manage. Exploring using Jupyter is like SSH if SSH had a full-fledged console.

0.1.1 SSH with Paramiko

Connect with Paramiko

```
client = paramiko.SSHClient()
client.set_missing_host_key_policy(
    paramiko.client.WarningPolicy)
client.connect("localhost", **connect_params)
```

You need to somehow collect the parameters: the host, and maybe some other connection details (keys, agent, etc.). Once you have those, connection is

easy. You can also configure the host keys more carefully, if there is a reasonable way to collect or save them.

0.1.2 Run a command

Run Command with Paramiko

```
res = client.exec_command("ls")
files = res[1].read().decode("ascii").splitlines()
files
```

```
['some_file', 'another_file']
```

Running a command can be done with exec_command. You could print out res[1] directly. In this case, with two files, eyeballing them is enough.

In general, it is nice to be able to run post-processing on the command output locally. No need for long pipelines with ssh or, worse, local remote pipelines!

0.1.3 Loop

Automate with Paramiko

```
files = set()
for connect_params in connect_params_list:
    client.connect("localhost", **connect_params)
    res = client.exec_command("ls")
    files.update(
        res[1].read().decode("ascii").splitlines()
    )
pprint.pprint(sorted(files), width=40)
```

```
['another_file',
 'even_more_files',
 'more_file',
 'some_file']
```

The real power shows up when you want to run the same command (or closely related commands) across a set of machines. For simplicity, here we consolidate all files into one list. This technique could be used to find which server has a core dump, for example, or restart a systemd service across several servers.

0.2 Cloud

Cloud

Why not...
Web UI?
Command-line?
Scripts?

There are many ways to interact with the cloud. You can use the web UI. This is fine, but unpleasant to automate.

You can use a custom command-line tool. There is nothing wrong with that either!

Most clouds, thoguh, have a well-supported Python library to automate them. Many of them even support a completely compatible AWS S3 interface, allowing you to manipulate remote static objects with the boto3 library.

0.2.1 Configuring `boto3`

Connect to S3

```
s3 = boto3.client(
    service_name='s3',
    region_name='us-west-2',
    # Credentials can be read from
    # different sources.
    **access_credentials,
)
```

Configuring the client can be tricky, and depends on specific details. There might be a local abstraction to get medium-lived credentials. Alternatively, you can run Jupyter on a cloud machine which gets credentials using a cloud-specific mechanism.

Since the notebooks are designed to be shared, include dynamic code that reads any secrets from local files, or accept them using a Jupyter input widget. Avoid putting tokens directly in the notebook.

0.2.2 Uploading an S3 file

Upload to S3

```
some_contents = io.BytesIO(b"some contents")
s3.upload_fileobj(
    some_contents,
    "special-bucket.123.431",
    "some-contents.txt",
)
```

One nice thing about using Jupyter here is that the *contents* of the files can be generated from code. In either case, having the logic for uploading the object to the store in a notebook means it can be repeated or tweaked.

0.2.3 Looping

Automate Uploading to S3

```
for i in range(10):
    some_contents = io.BytesIO(
```

```
        f"some {i} contents".encode("ascii")
    )
    s3.upload_fileobj(
        some_contents,
        "special-bucket.123.431",
        f"some-contents-{i}.txt",
    )
```

Even nicer, you can *loop*. Upload multiple objects to the bucket, or upload the same object to multiple buckets. This is useful, for example, if you need to modify an object-store-backed website or several websites in some automated way.

Beyond that, though other examples are less trivial to find on a slide, you can do other kinds of cloud-based automation using these ideas.

0.3 Source collaboration platforms

Source collaboration platforms

Examples: GitHub, GitLab, BitBucket, ...

GitLab: open core

Why? Multi-repo management!

It is the year 2022 CE. Most of us use git-backed web-based source collaboration platforms. The most well-known is GitHub, but GitLab and BitBucket are famous examples.

The following examples show-case GitLab. The basic ideas will be similar for all of them, but GitLab's open core model means you can test these out against a local installation fo GitLab using only the open source version.

0.3.1 Configuring the client

Configuring Gitlab

```
client = gitlab.Gitlab(private_token=token)
```

You want to avoid using your username/password. All of the popular platforms enable a web-based flow to collect a private access token (optionally with an expiration date).

As before, avoid embedding the token directly into the notebook.

0.3.2 Analyzing one project

Analyzing README

```
project = client.projects.get(project_name)
[readme] = [
    obj
    for obj in project.repository_tree(
        as_list=False
```

```
    )
    if obj["name"] == "README.md"
]
contents = project.repository_blob(readme["id"])
data = base64.b64decode(
    contents["content"].encode("ascii")
).decode("utf-8")
len(data.split())
```

```
882
```

In the interest of having an example that fits on a single slide, this is a simple README word count. As an example of a real use case, imagine that you want to estimate reading times for various READMEs.

0.3.3 Looping

Analyzing projects in a loop

```
for project_name in projects:
    project = client.projects.get(project_name)
    [readme] = [
        obj
        for obj in project.repository_tree(
            as_list=False
        )
        if obj["name"] == "README.md"
    ]
    contents = project.repository_blob(readme["id"])
    data = base64.b64decode(
        contents["content"].encode("ascii")
    ).decode("utf-8")
    print(len(data.split()))
```

```
882
563
```

After checking that the code flow, which can be non-trivial, works on one project, looping is the next. This requires indenting the code from the previous slide, and adding a for loop on top.

This can be added to a scanner that automatically warns if the README is either too short (probably needs more) or too long (and needs to be broken up).

0.4 Summary

Use Jupyter for DevOps

- Prototype
- Iterate
- Automate
- Document
- Share

Try it yourself! Install Jupyter and use it for day to day tasks. Prototype automation or remediation tasks, iterate on them, and then run them on all relevant systems, without switching windows!

Jupyter can also be used to document what you did: useful as an attachment to a break-fix ticket. Expore notebooks to HTML, after adding some markdown cells, and you have something you can send to colleagues to help with knowledge sharing – even if these colleagues do not use Jupyter (yet!).