Automation for Your Application on Code Changes

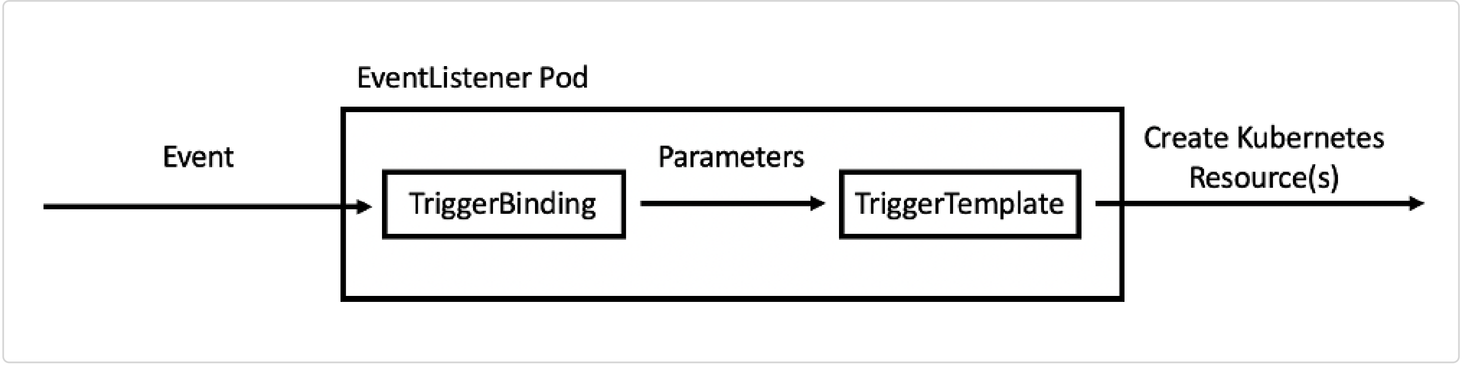
Background: Web Hooks

Most Git repository servers support the concept of web hooks — calling to an external source via HTTP(S) when a change in the code repository happens. OpenShift provides an API endpoint that supports receiving hooks from remote systems in order to trigger builds. By pointing the code repository's hook at the OpenShift Pipelines resources, automated code/build/deploy pipelines can be achieved.

Adding Triggers to your Pipeline

Tekton **Triggers** enable us to configure Pipelines to respond to external events (Git push events, pull requests etc) such as Web Hooks.

Adding triggering support requires the creation of a `TriggerTemplate`, `TriggerBinding`, and an `EventListener` in our project.



Let’s see each component in detail:

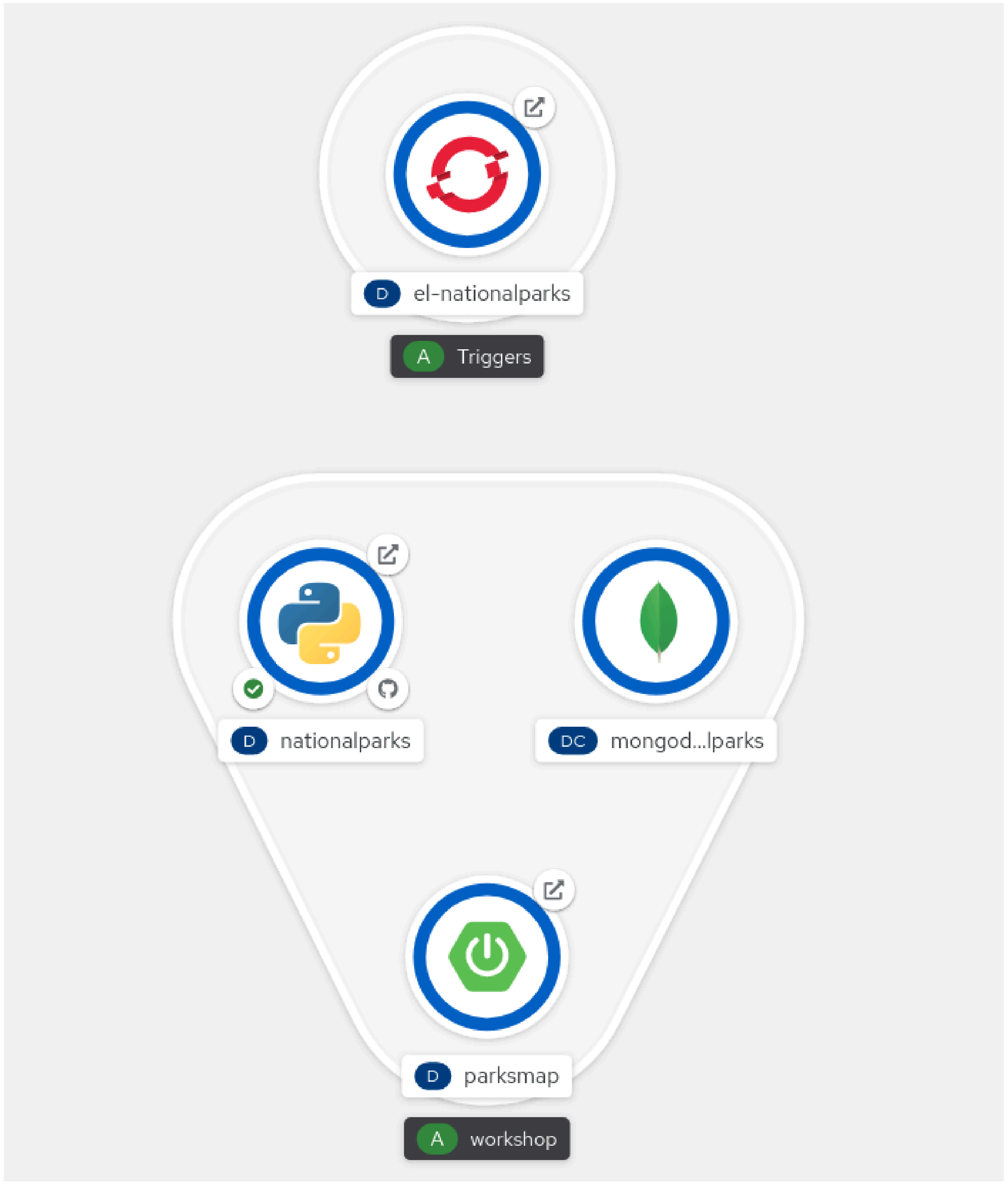
- **TriggerTemplate**: a trigger template is a template for newly created resources. It supports parameters to create specific `PipelineResources` and `PipelineRuns`.
- **TriggerBinding**: validates events and extracts payload fields
- **EventListener**: connects `TriggerBindings` and `TriggerTemplates` into an addressable endpoint (the event sink). It uses the extracted event parameters from each TriggerBinding (and any supplied static parameters) to create the resources specified in the corresponding TriggerTemplate. It also optionally allows an external service to pre-process the event payload via the interceptor field.

Now let’s create them all together for our Pipeline:

```
oc create -f http://gogs-labs.apps.rosa-7s42b.rfax.p1.openshiftapps.com/user4/nationalparks-py/raw/master/pipelines/nationalparks-triggers-all.yaml -n user4
```

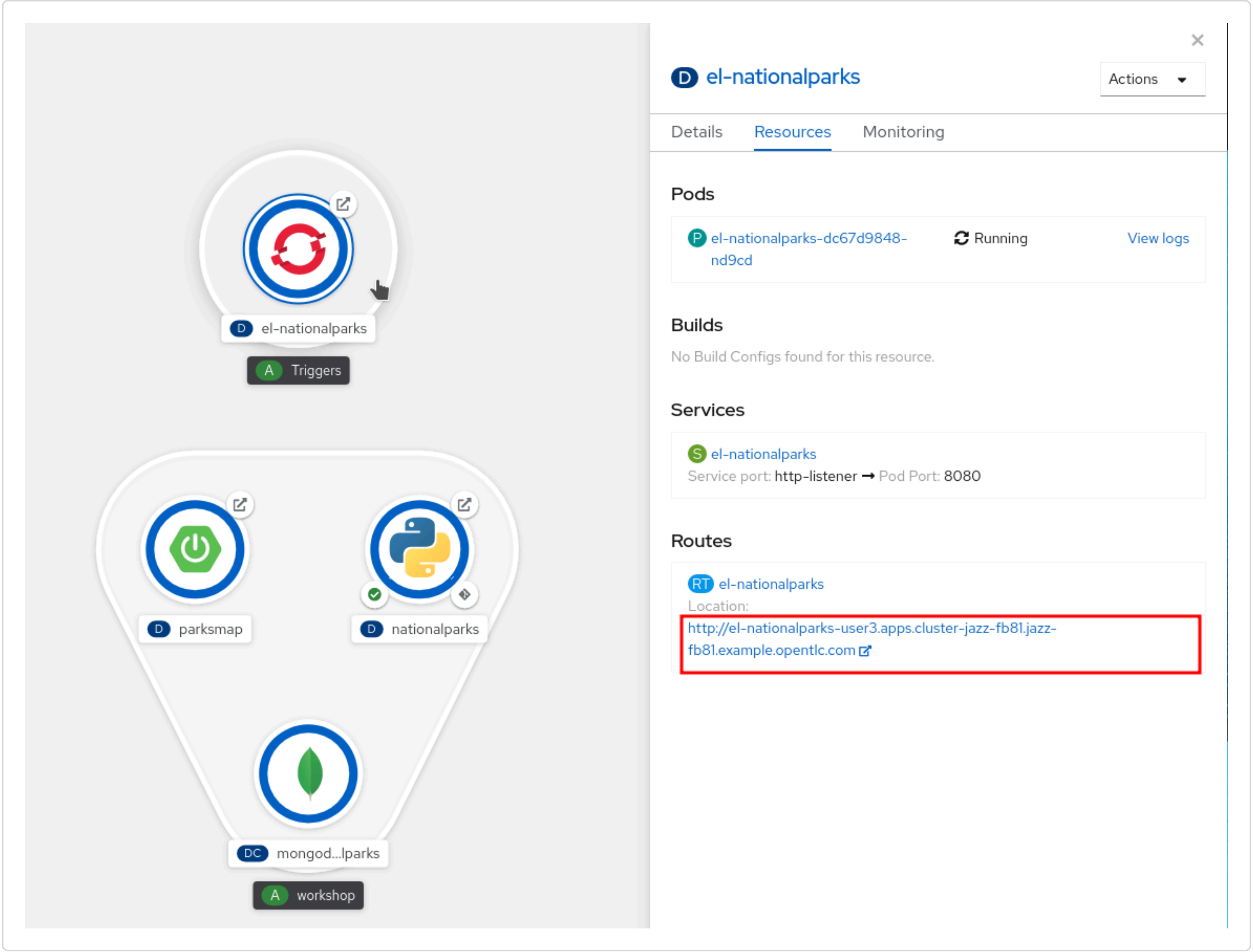
This will create a new Pod with a Route that we can use to setup our Webhook on Gogs to trigger the automatic start of the Pipeline.

From left side menu, click on **Topology** to verify if a new Deployment **el-nationalparks** for the `EventListener` has ben created:


D el-nationalparks
A Triggers
D nationalparks
DC mongod...lparks
D parksmap
A workshop

Exercise: Configuring Gogs Web Hooks

Click on the Deployment, go into **Routes** section and and copy the **el-nationparks** Route URL.



D el-nationalparks

Pods

P el-nationalparks-dc67d9848-nd9cd Running View logs

Builds

No Build Configs found for this resource.

Services

S el-nationalparks
Service port: http-listener → Pod Port: 8080

Routes

RT el-nationalparks
Location:
http://el-nationalparks-user3.apps.cluster-jazz-fb81.jazz-fb81.example.opentlc.com

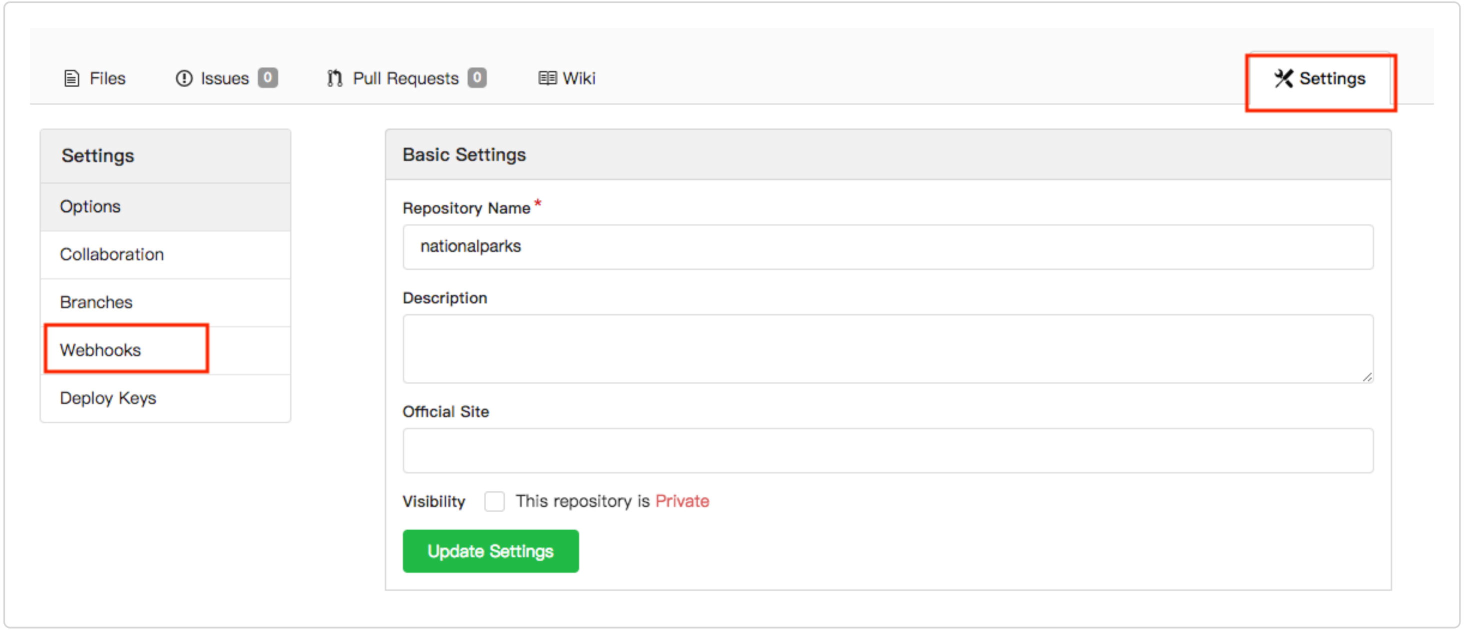
Once you have the URL copied to your clipboard, navigate to the code repository that you have on Gogs:

Gogs Repository

Your Gogs credentials are:

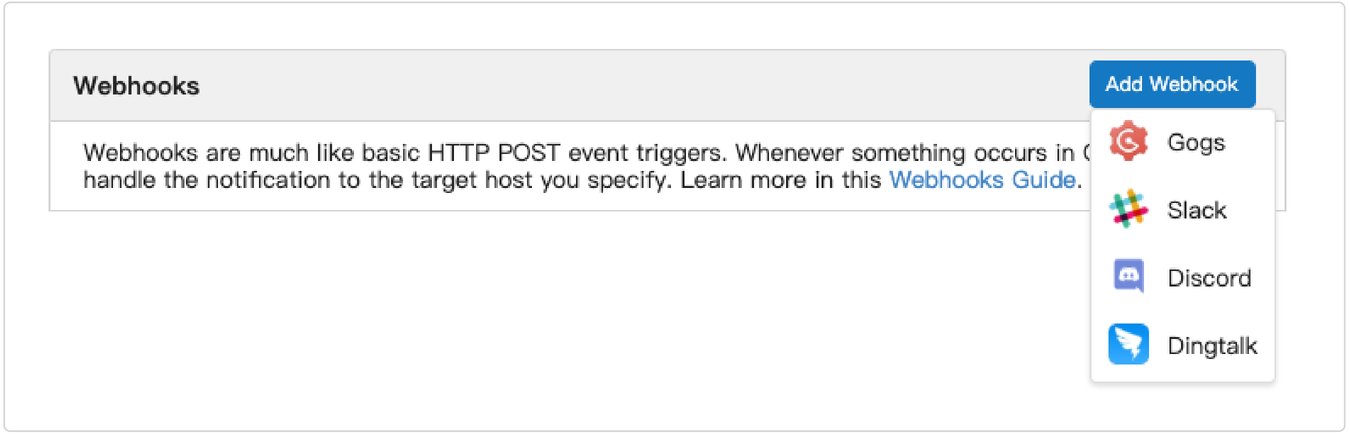
```
username: user4
password: gogs
```

Click the Settings link on the top right of the screen:



Repository Name *
nationalparks
Description
Official Site
Visibility This repository is Private
Update Settings

Click on **Webhooks**, then the **Add Webhook** button, and finally select **Gogs**.

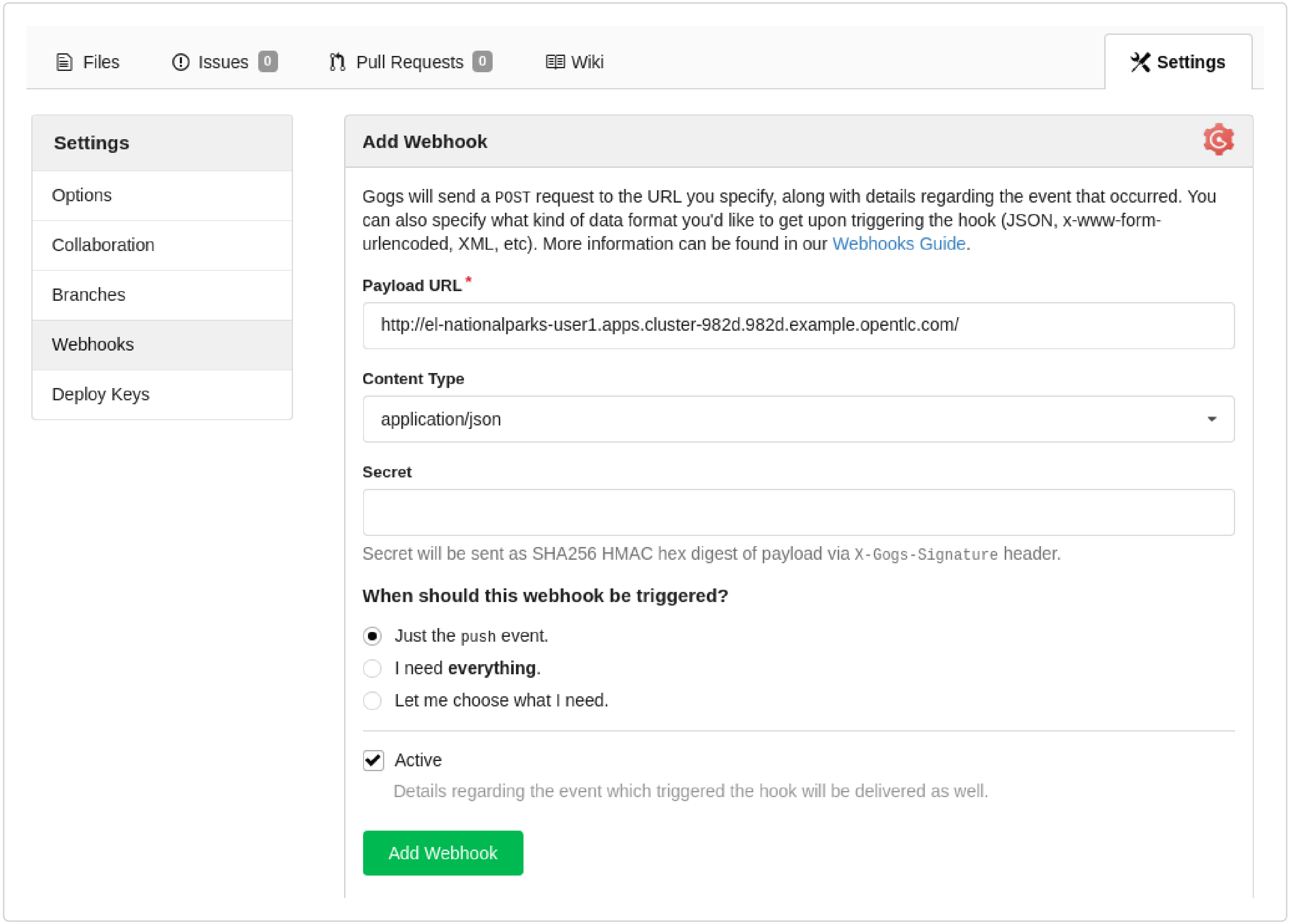


Webhooks are much like basic HTTP POST event triggers. Whenever something occurs in (
handle the notification to the target host you specify. Learn more in this Webhooks Guide.

In the next screen, paste your link into the "Payload URL" field. You can leave the secret token field blank — the secret is already in the URL and does not need to be in the payload.

Change the `Content Type` to `application/json`.

Finally, click on **Add Webhook**.



Add Webhook

Gogs will send a `POST` request to the URL you specify, along with details regarding the event that occurred. You can also specify what kind of data format you'd like to get upon triggering the hook (JSON, x-www-form-urlencoded, XML, etc). More information can be found in our Webhooks Guide.

Payload URL *
http://el-nationalparks-user1.apps.cluster-982d.982d.example.opentlc.com/

Content Type
application/json

Secret

Secret will be sent as SHA256 HMAC hex digest of payload via `X-Gogs-Signature` header.

When should this webhook be triggered?

- ● Just the `push` event.
- ○ I need **everything**.
- ○ Let me choose what I need.

✓ Active
Details regarding the event which triggered the hook will be delivered as well.

Add Webhook

Boom! From now on, every time you commit new source code to your Gogs repository, a new build and deploy will occur inside of OpenShift. Let's try this out.

Exercise: Using Gogs Web Hooks

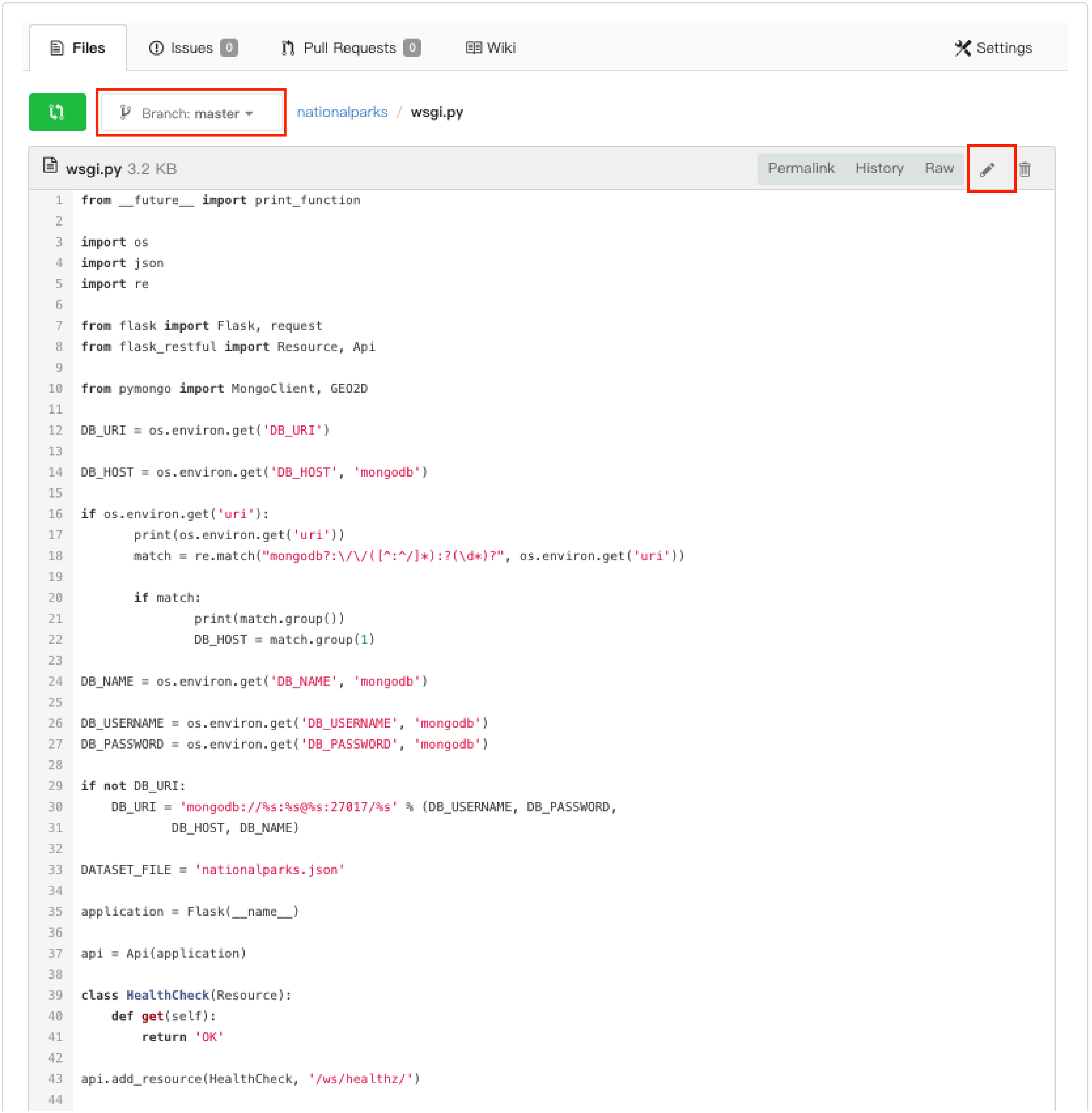
Click the **Files** tab in Gogs. This is Gogs's repository view.

> Make sure that the drop-down menu at the upper right is set for the `master` branch. Navigate to the following path:

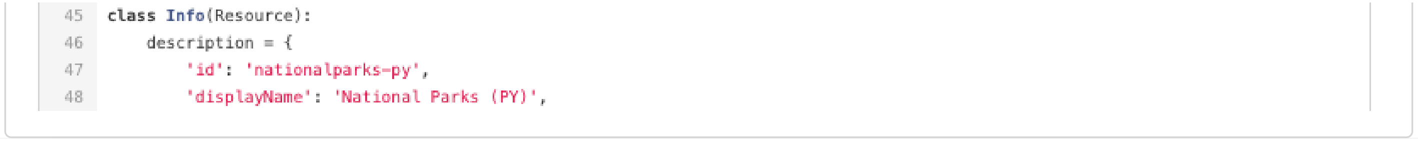
```
/
```

Then click on the `wsgi.py` file.

Once you have the file on the screen, click the edit button in the top right hand corner as shown here:



```
from __future__ import print_function

import os
import json
import re

from flask import Flask, request
from flask_restful import Resource, Api

from pymongo import MongoClient, GEO2D

DB_URI = os.environ.get('DB_URI')

DB_HOST = os.environ.get('DB_HOST', 'mongodb')

if os.environ.get('uri'):
        print(os.environ.get('uri'))
        match = re.match("mongodb?:\/\/([^:^/]*):?(\d*)?", os.environ.get('uri'))

        if match:
                print(match.group())
                DB_HOST = match.group(1)

DB_NAME = os.environ.get('DB_NAME', 'mongodb')

DB_USERNAME = os.environ.get('DB_USERNAME', 'mongodb')
DB_PASSWORD = os.environ.get('DB_PASSWORD', 'mongodb')

if not DB_URI:
    DB_URI = 'mongodb://%s:%s@%s:27017/%s' % (DB_USERNAME, DB_PASSWORD,
            DB_HOST, DB_NAME)

DATASET_FILE = 'nationalparks.json'

application = Flask(__name__)

api = Api(application)

class HealthCheck(Resource):
    def get(self):
        return 'OK'

api.add_resource(HealthCheck, '/ws/healthz/')

```



```
class Info(Resource):
    description = {
        'id': 'nationalparks-py',
        'displayName': 'National Parks (PY)',
```

Change line number 50:

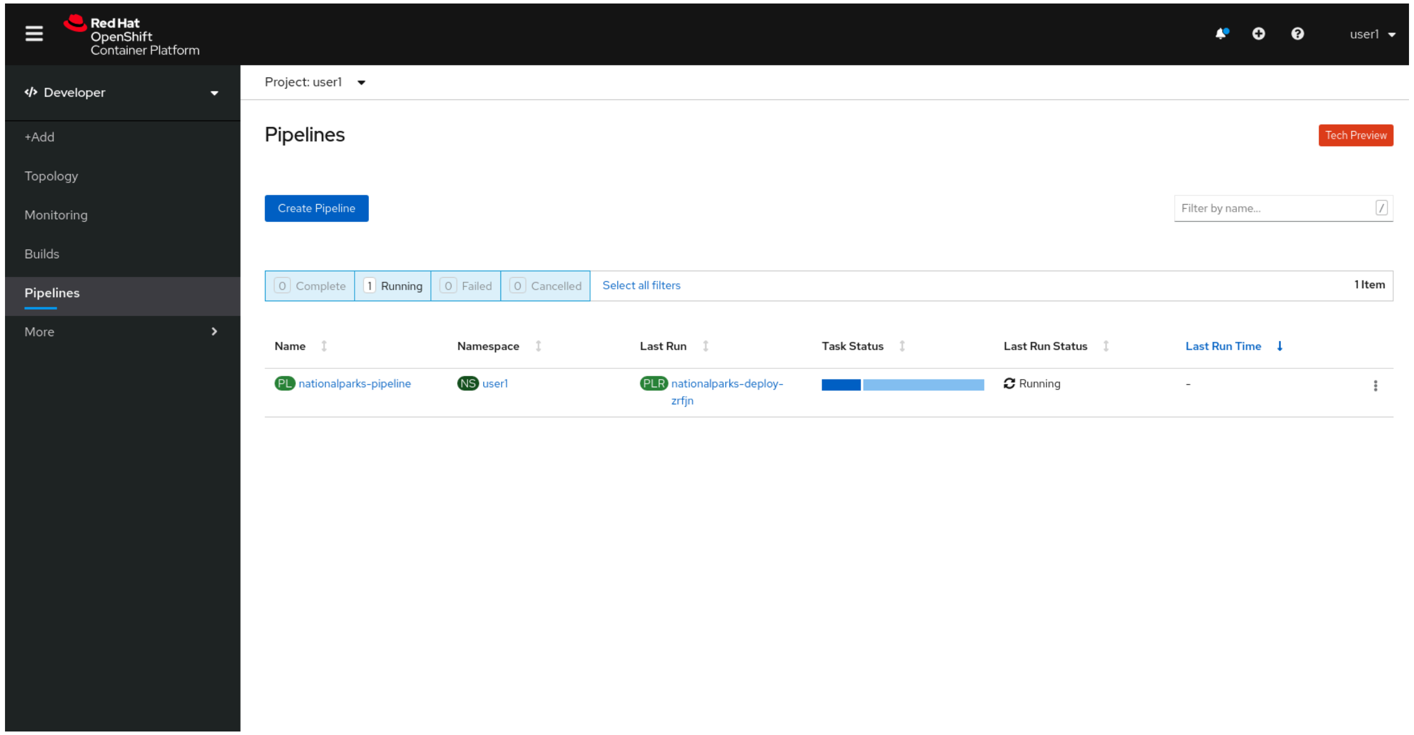
```
'displayName': 'National Parks (PY)',
```

To

```
'displayName': 'Amazing National Parks (PY)',
```

Click on Commit changes at the bottom of the screen. Feel free to enter a commit message.

Once you have committed your changes, a new **PipelineRun** should almost instantaneously be triggered in OpenShift. Click **Pipeline** in the left navigation menu then `nationalparks-pipeline`. You should see a new one running:



Name	Namespace	Last Run	Task Status	Last Run Status	Last Run Time
PL nationalparks-pipeline	NS user1	PLR nationalparks-deploy-zrfjn		Running	-

or run the following command to verify:

```
oc get PipelineRun
```

Once the build and deploy has finished, verify your new image was automatically deployed by viewing the application in your browser:

National Parks Info Page

You should now see the new name you have set in the JSON string returned.

> To see this in the map’s legend itself, you will need to scale down your parksmap to 0, then back up to 1 to force the app to refresh its cache.

Continue