CI/CD PW Workflow

Github Actions and PW Client

Introduction

The setup steps described here have already been done and are left in this account as a template. Please feel free to replicate these steps to get a feel for how they work.

Where is the source code?

1. weather-cluster-demo: repository with weather model install and launch scripts
2. test-workflow-action: repository containing the GitHub action linked to weather-cluster-demo.
3. beta.parallel.works: PW SaaS platform for trial. Please login here with PW credentials. The code is already on the platform in a workflow directory in `/pw/workflows/weather-cluster-demo`.

TODO:

- Add connectivity driver information (no need to setup EFA/gVINC yourself)
- Add hpc6a and build/test everything on Azure

Introduction

Where is the documentation?

1. weather-cluster-demo/README.md: Software installation and how to run the weather model application.
2. test-workflow-action/README.md: using the GitHub action
3. This slide deck: Summary of the steps to setup the model launched by action.
4. PW platform ? buttons will open a new tab (but slightly out of date).

Where to start?

1. Log in to PW to view the resource configurations, IDE, starting/stopping clusters, and interactive access to `*.clusters.pw`.
2. The GitHub action can be run directly from weather-cluster-demo on GitHub - the same weather model will be run on atNorth, AWS, and GCE.
3. **On PW,** `/pw/workflows/weather-cluster-demo/main.sh` is the core code launched by the workflow; it will clone the repo, launch the model on the clusters, and monitor the status of the application.

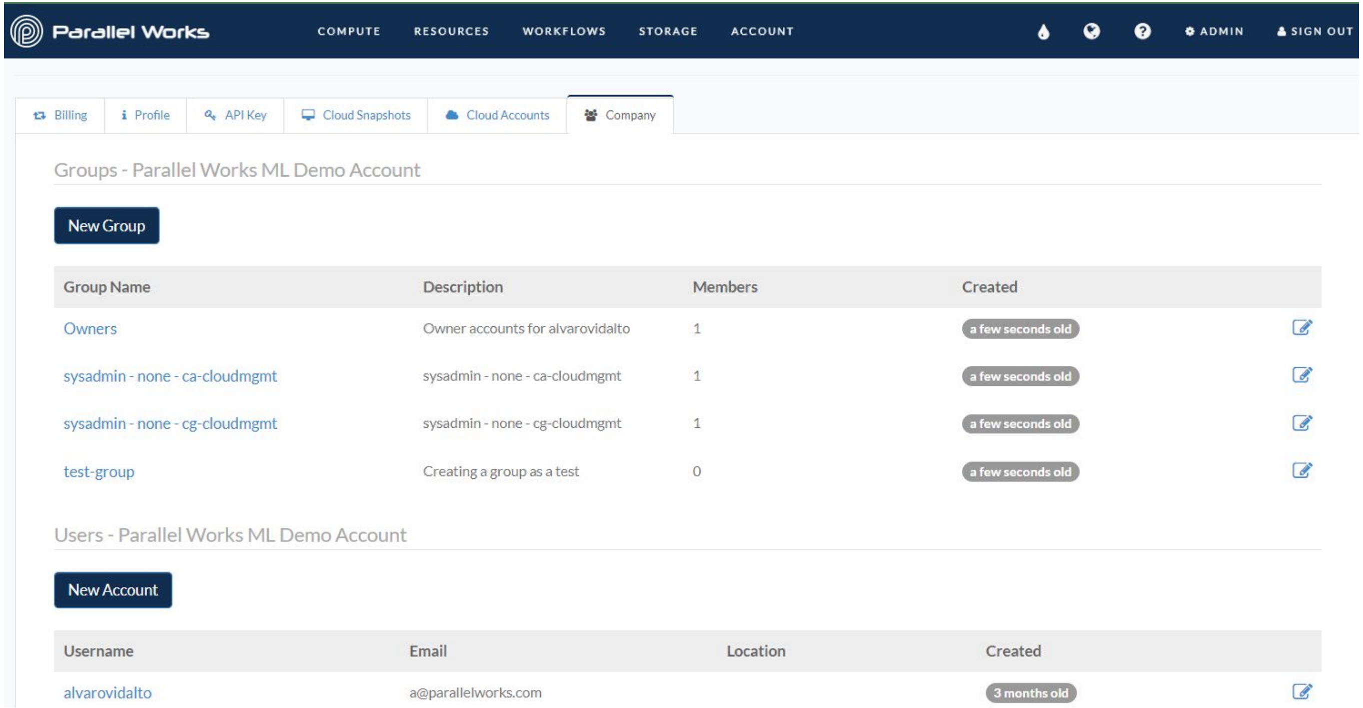
Introduction

Where are things on the clusters?

1. The WRF application code is in `/var/lib/pworks/spack` on GCE and AWS. atNorth, the application code is in `/shared/wrf/spack`.
2. GCE and AWS clusters share `$HOME` between the head node and worker nodes, so the working directory for WRF is in `$HOME/weather-forecast-demo/<jobid>/weather-forecast-demo/conus_12km` . Initial setup of this working directory is done with `local_setup.sh` because `$HOME` is not persistent (i.e. not in a cloud disk or in the image).
3. atNorth clusters do not share `$HOME` between head node and worker nodes and the `$HOME`(s) are persistent. Instead, the working directory is in `/shared/weather-forecast-demo/<jobid>/weather-forecast-demo/conus_12km` .
4. In all cases, you can track the WRF run via the main log file `conus<job>*.out`, and the 0 rank MPI process' log `rsl.error.0000`.
5. Output is in NetCDF format in `wrfout*` files.

Setup step 1:
Setup projects

Projects do **NOT** apply to atNorth



Groups - Parallel Works ML Demo Account

Group Name	Description	Members	Created
Owners	Owner accounts for alvarovidalto	1	a few seconds old
sysadmin - none - ca-cloudmgmt	sysadmin - none - ca-cloudmgmt	1	a few seconds old
sysadmin - none - cg-cloudmgmt	sysadmin - none - cg-cloudmgmt	1	a few seconds old
test-group	Creating a group as a test	0	a few seconds old

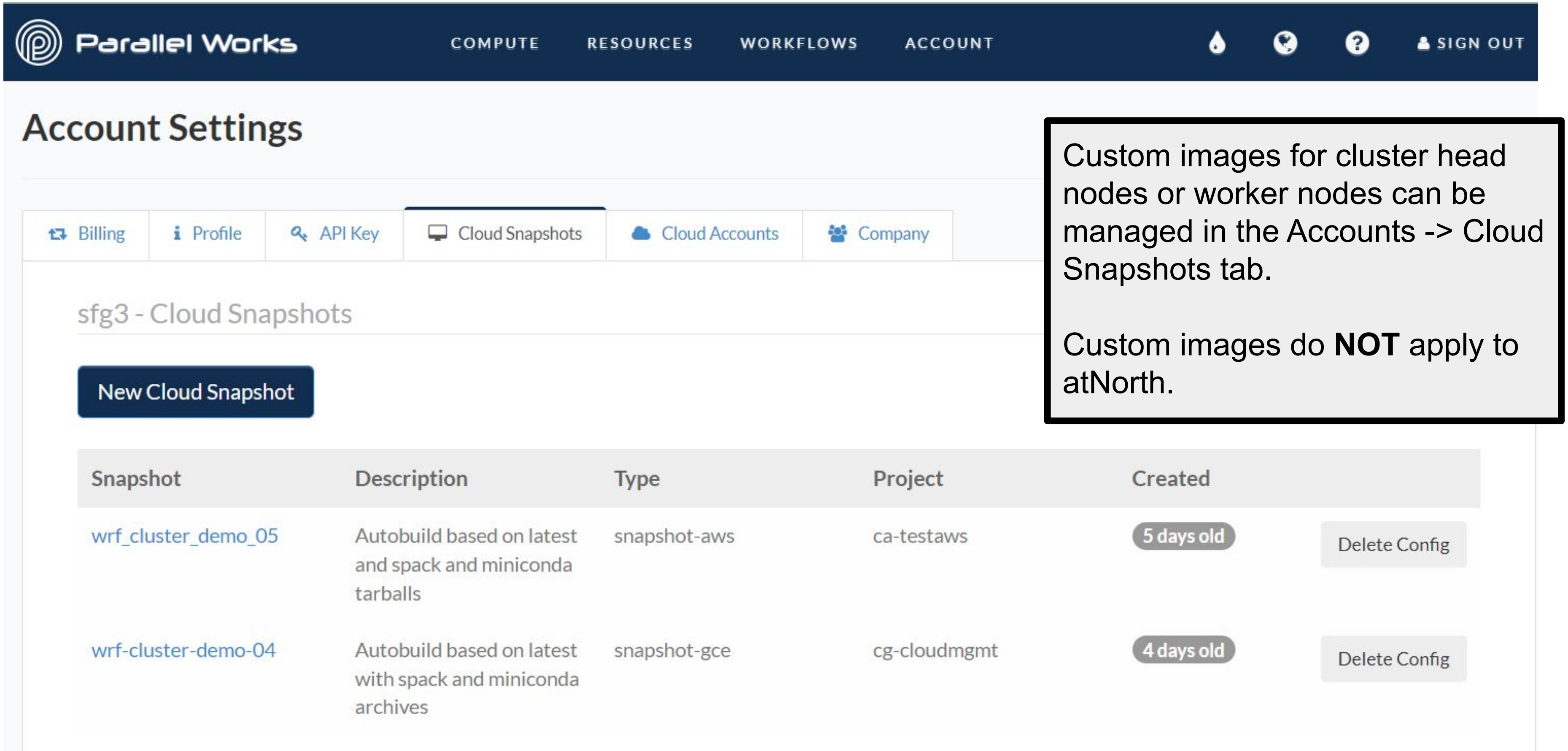
Users - Parallel Works ML Demo Account

Username	Email	Location	Created
alvarovidalto	a@parallelworks.com		3 months old

PW “main” user accounts can create subaccounts and groups of accounts in the Account -> Company tab. These groups are the “projects” used in the cluster configuration step, later. To use a group/project, simply add a user to it. This applies to main user accounts and subaccounts. Currently, projects for the major cloud providers require the following prefixes:

- GCE: cg-<project_name>
- AWS: ca-<project_name>
- Azure: cz-<project_name>

Setup step 2a: Manage images



Account Settings

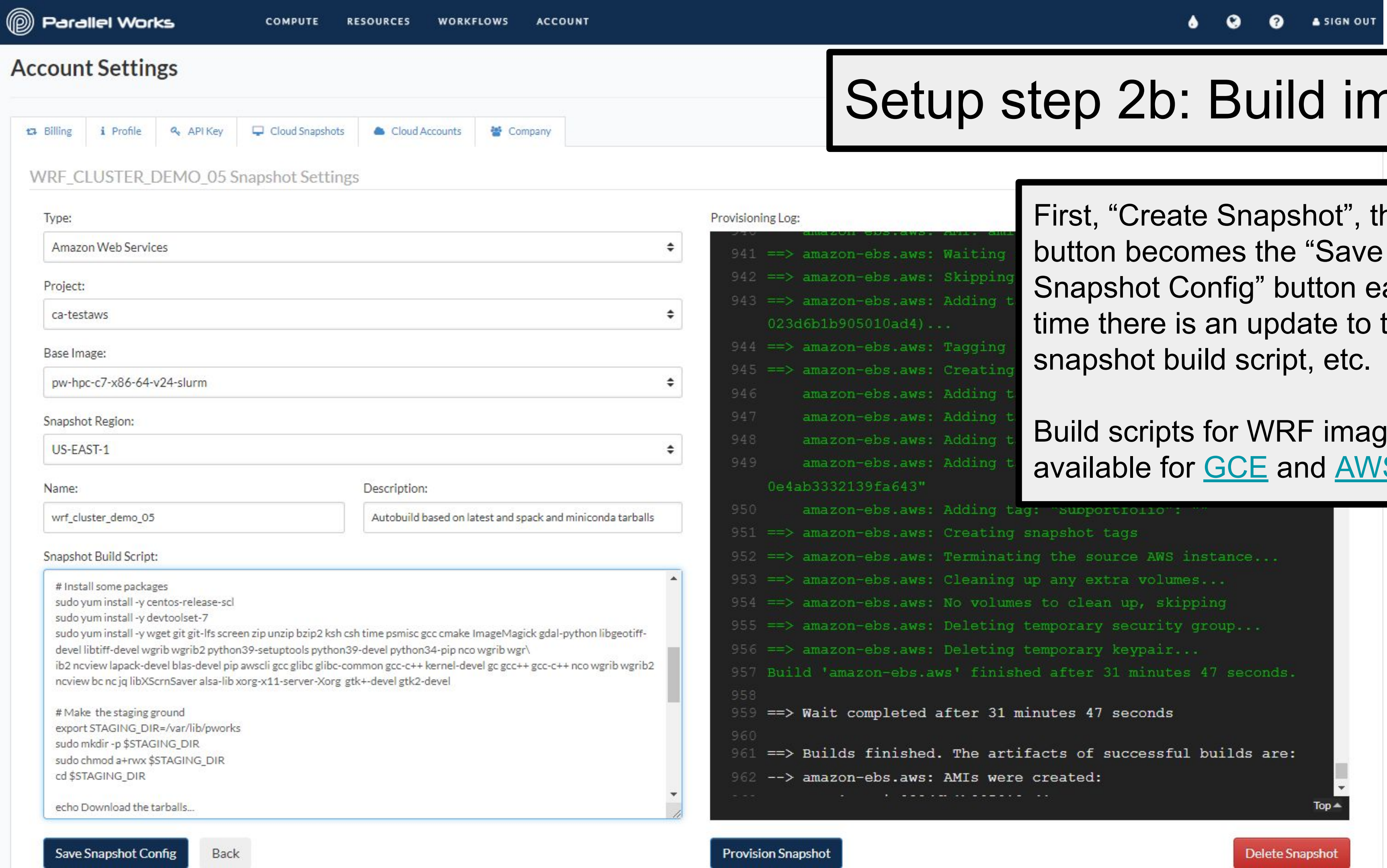
sfg3 - Cloud Snapshots

Snapshot	Description	Type	Project	Created	
wrf_cluster_demo_05	Autobuild based on latest and spack and miniconda tarballs	snapshot-aws	ca-testaws	5 days old	Delete Config
wrf-cluster-demo-04	Autobuild based on latest with spack and miniconda archives	snapshot-gce	cg-cloudmgmt	4 days old	Delete Config

Custom images for cluster head nodes or worker nodes can be managed in the Accounts -> Cloud Snapshots tab.

Custom images do **NOT** apply to atNorth.

Setup step 2b: Build images



Account Settings

WRF_CLUSTER_DEMO_05 Snapshot Settings

Type: Amazon Web Services

Project: ca-testaws

Base Image: pw-hpc-c7-x86-64-v24-slurm

Snapshot Region: US-EAST-1

Name: wrf_cluster_demo_05

Description: Autobuild based on latest and spack and miniconda tarballs

Snapshot Build Script:

```
# Install some packages
sudo yum install -y centos-release-scl
sudo yum install -y devtoolset-7
sudo yum install -y wget git git-lfs screen zip unzip bzip2 ksh csh time psmisc gcc cmake ImageMagick gdal-python libgeotiff-devel libtiff-devel wgrib wgrib2 python39-setuptools python39-devel python34-pip nco wgrib wgr\
ib2 ncview lapack-devel blas-devel pip awscli gcc glibc glibc-common gcc-c++ kernel-devel gc gcc++ gcc-c++ nco wgrib wgrib2 ncview bc nc jq libXScrnSaver alsa-lib xorg-x11-server-Xorg  gtk+-devel gtk2-devel

# Make  the staging ground
export STAGING_DIR=/var/lib/pworks
sudo mkdir -p $STAGING_DIR
sudo chmod a+rwx $STAGING_DIR
cd $STAGING_DIR

echo Download the tarballs...
```

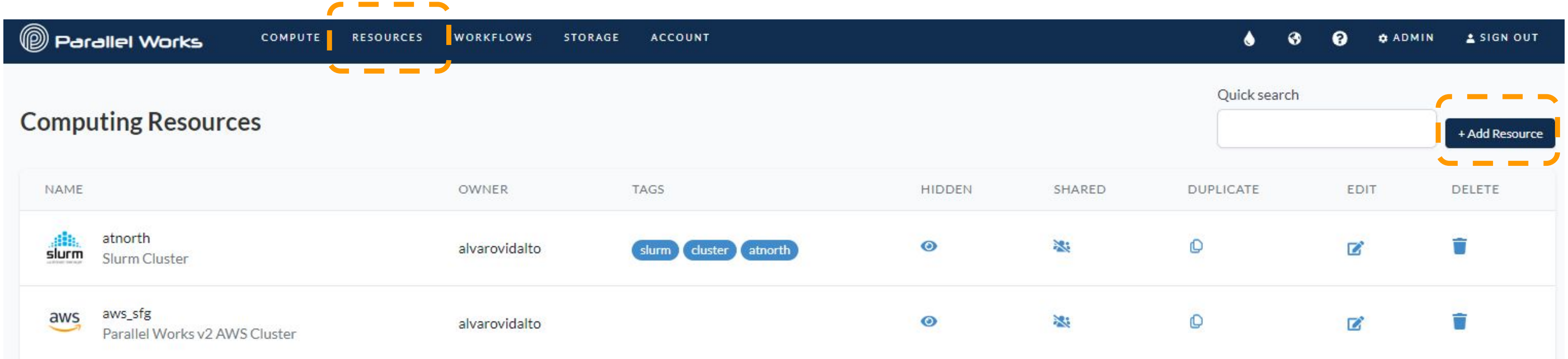
Provisioning Log:

```
==> amazon-ebs.aws: Waiting
==> amazon-ebs.aws: Skipping
==> amazon-ebs.aws: Adding t
023d6b1b905010ad4)...
==> amazon-ebs.aws: Tagging
==> amazon-ebs.aws: Creating
    amazon-ebs.aws: Adding t
    amazon-ebs.aws: Adding t
    amazon-ebs.aws: Adding t
    amazon-ebs.aws: Adding t
0e4ab3332139fa643"
    amazon-ebs.aws: Adding tag: "Subportfolio": ""
==> amazon-ebs.aws: Creating snapshot tags
==> amazon-ebs.aws: Terminating the source AWS instance...
==> amazon-ebs.aws: Cleaning up any extra volumes...
==> amazon-ebs.aws: No volumes to clean up, skipping
==> amazon-ebs.aws: Deleting temporary security group...
==> amazon-ebs.aws: Deleting temporary keypair...
Build 'amazon-ebs.aws' finished after 31 minutes 47 seconds.

==> Wait completed after 31 minutes 47 seconds

==> Builds finished. The artifacts of successful builds are:
--> amazon-ebs.aws: AMIs were created:
```

First, “Create Snapshot”, then that button becomes the “Save Snapshot Config” button each time there is an update to the snapshot build script, etc.

Build scripts for WRF images are available for GCE and AWS.

Setup step 3a: Manage resources



Computing Resources

NAME	OWNER	TAGS	HIDDEN	SHARED	DUPLICATE	EDIT	DELETE
atnorth Slurm Cluster	alvarovidalto	slurm cluster atnorth					
aws_sfg Parallel Works v2 AWS Cluster	alvarovidalto						

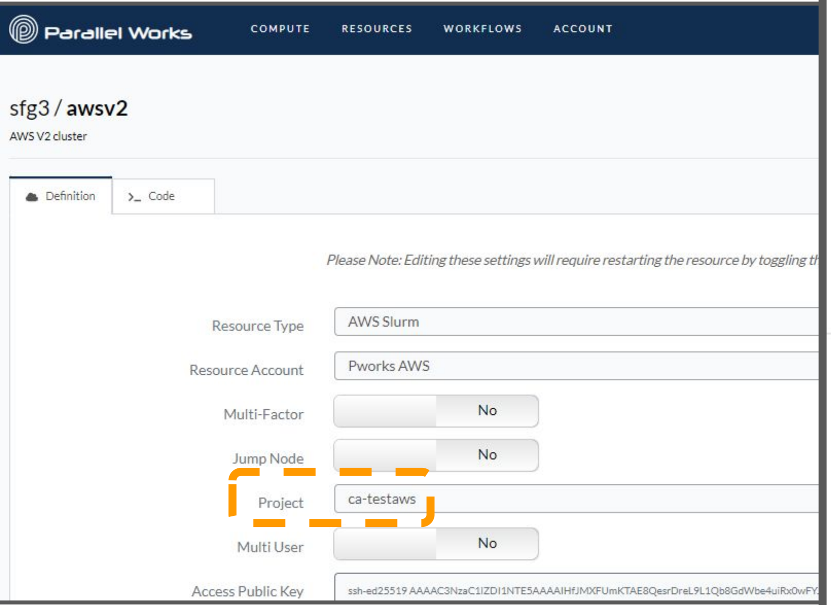
After logging in, go to the Resources Tab and select either Add Resource or an existing resource. There are three types of resources:

1. persistent clusters, e.g. atNorth (“Slurm Cluster” provider)
2. **cloud clusters (please use V2 clusters for this trial)**
3. worker pools (workers nodes are independent, no head node)

When configuring a new cluster,

1. Select a project created in step 1
2. Select an image created in step 2
3. Select the compute resources of interest. **Note:** AWS hpc6a instances are only available in us-east-2.

Setup step 3b: Configure resources

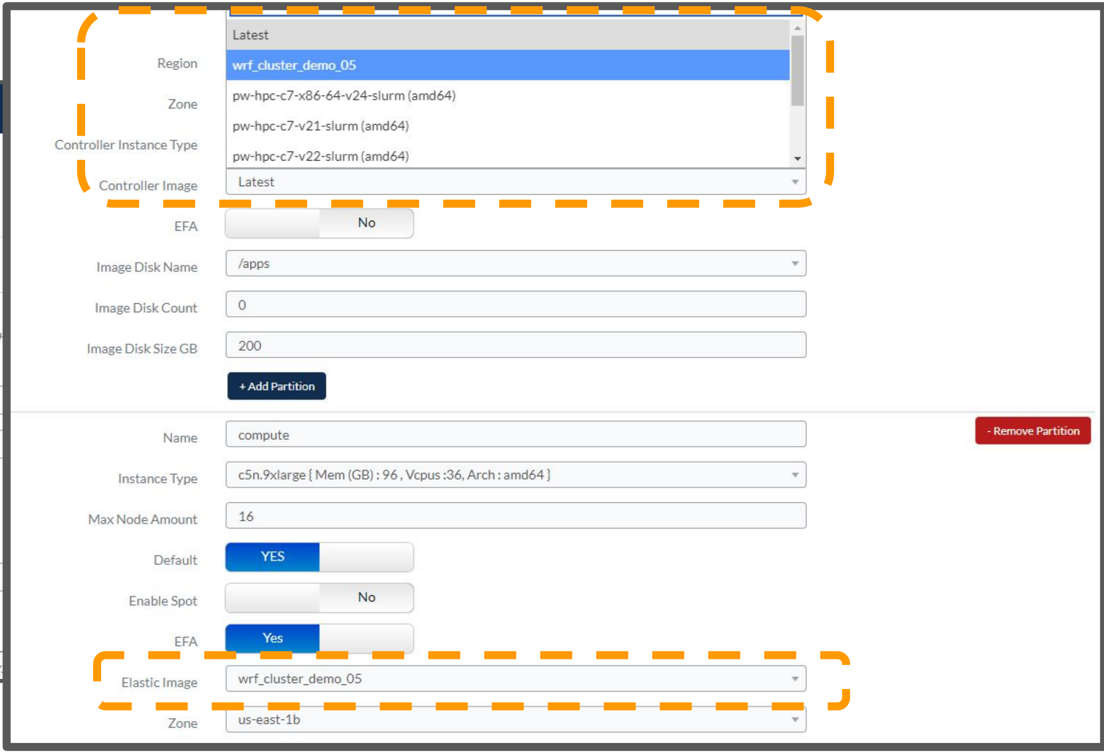


sfg3 / awsv2

AWS V2 cluster

Please Note: Editing these settings will require restarting the resource by toggling th

Resource Type	AWS Slurm
Resource Account	Pworks AWS
Multi-Factor	No
Jump Node	No
Project	ca-testaws
Multi User	No
Access Public Key	ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIHfJMXFUmKTAE8QesrDreL9L1Qb8GdWbe4uiRx0wFY



Latest
wrf_cluster_demo_05
pw-hpc-c7-x86-64-v24-slurm (amd64)
pw-hpc-c7-v21-slurm (amd64)
pw-hpc-c7-v22-slurm (amd64)

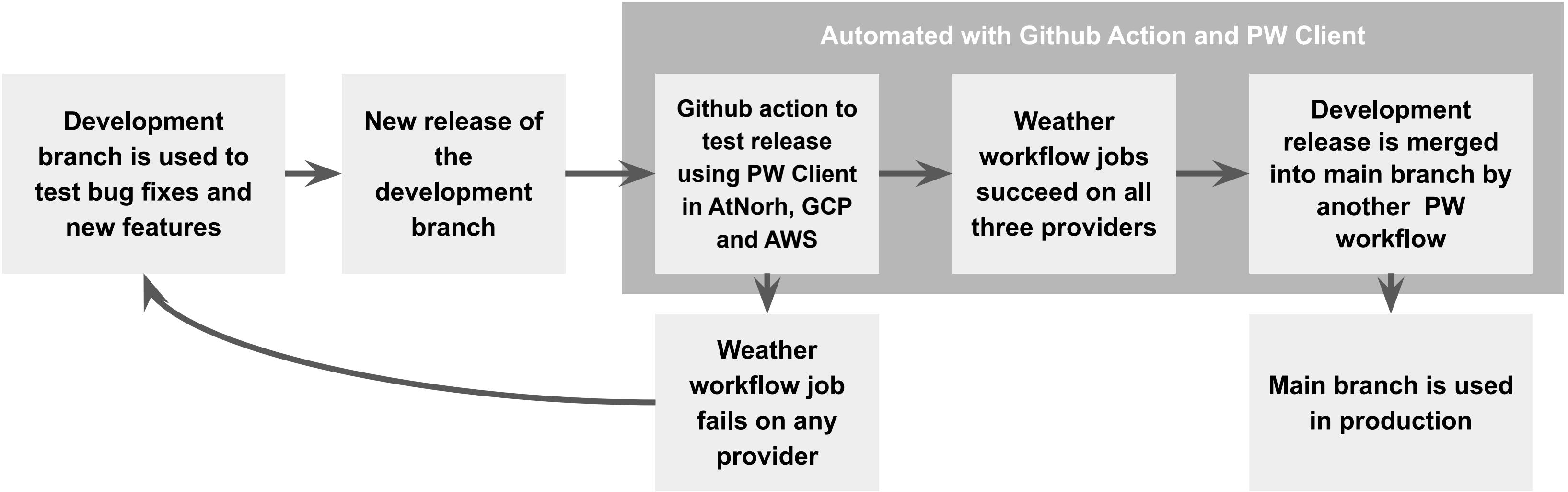
Region	
Zone	
Controller Instance Type	
Controller Image	Latest
EFA	No
Image Disk Name	/apps
Image Disk Count	0
Image Disk Size GB	200
Name	compute
Instance Type	c5n.9xlarge { Mem (GB) : 96 , Vcpus :36, Arch : amd64 }
Max Node Amount	16
Default	YES
Enable Spot	No
EFA	Yes
Elastic Image	wrf_cluster_demo_05
Zone	us-east-1b

When configuring a new cluster,

1. Select a project created in step 1 (left figure)
2. Select an image created in step 2 (right figure) + select instances, etc.
3. Select the compute resources of interest. **Note:** AWS hpc6a instances are only available in us-east-2.

Example 1:

- PW workflow clones a Github repository at runtime (when a PW job is submitted)
- Github repository has two branches:
  - Main: Is cloned by default in production
  - Development: Used for development
- A Github action is used to test new releases of the development branch and merge them into the main branch
  - Github action uses PW Client to automate workflow execution across multiple resource providers
- Deploy keys are used to control read and write access to the repository
- Links to the test-workflow-action and its implementation in the weather demo repository

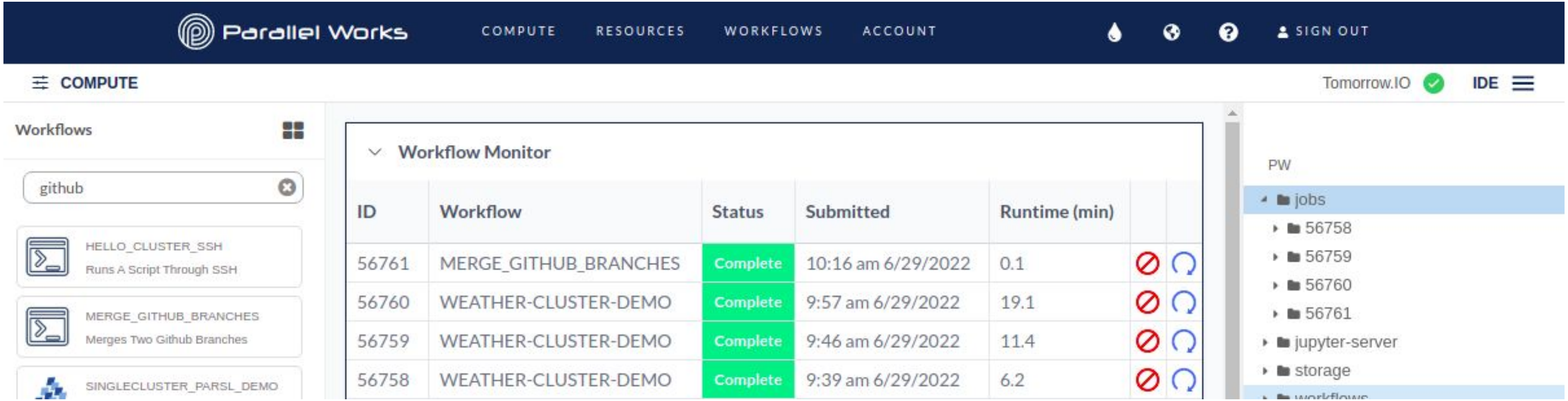

Automated with Github Action and PW Client
Development branch is used to test bug fixes and new features
New release of the development branch
Github action to test release using PW Client in AtNorh, GCP and AWS
Weather workflow jobs succeed on all three providers
Development release is merged into main branch by another PW workflow
Weather workflow job fails on any provider
Main branch is used in production

Example 1:

These are the 4 PW jobs launched by the action on the new development release:

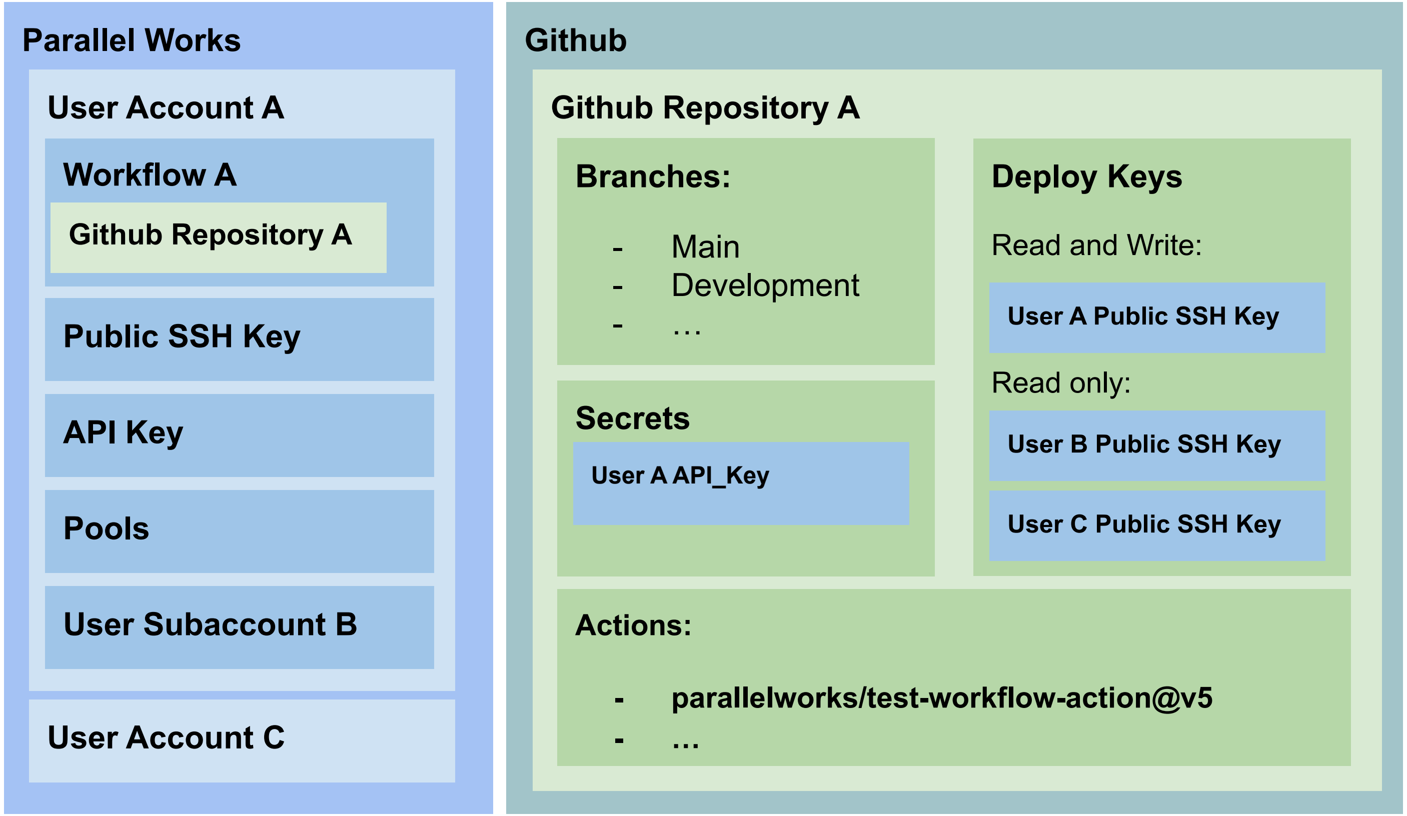
- **56758**: Testing the weather-cluster-demo workflow in AtNorth
- **56759**: Testing the weather-cluster-demo workflow in GCP
- **56760**: Testing the weather-cluster-demo workflow in AWS
- **56761**: Merging the development release into the merge branch with the merge_github_branches workflow

Job status is “Complete” if the exit code is 0 and “Error” otherwise. Error handling (including exit code) is up to the workflow developer (see /pw/workflows/weather-cluster-demo/main.sh)



ID	Workflow	Status	Submitted	Runtime (min)		
56761	MERGE_GITHUB_BRANCHES	Complete	10:16 am 6/29/2022	0.1	⊘	○
56760	WEATHER-CLUSTER-DEMO	Complete	9:57 am 6/29/2022	19.1	⊘	○
56759	WEATHER-CLUSTER-DEMO	Complete	9:46 am 6/29/2022	11.4	⊘	○
56758	WEATHER-CLUSTER-DEMO	Complete	9:39 am 6/29/2022	6.2	⊘	○

Key Components



Parallel Works

User Account A

Workflow A

Github Repository A

Public SSH Key

API Key

Pools

User Subaccount B

User Account C

Github

Github Repository A

Branches:

- Main
- Development
- …

Secrets

User A API_Key

Deploy Keys

Read and Write:

User A Public SSH Key

Read only:

User B Public SSH Key

User C Public SSH Key

Actions:

- **parallelworks/test-workflow-action@v5**
- **…**

Github action runs Workflow A in User Account A

- Need User API Key

Workflow A merges development branch into main

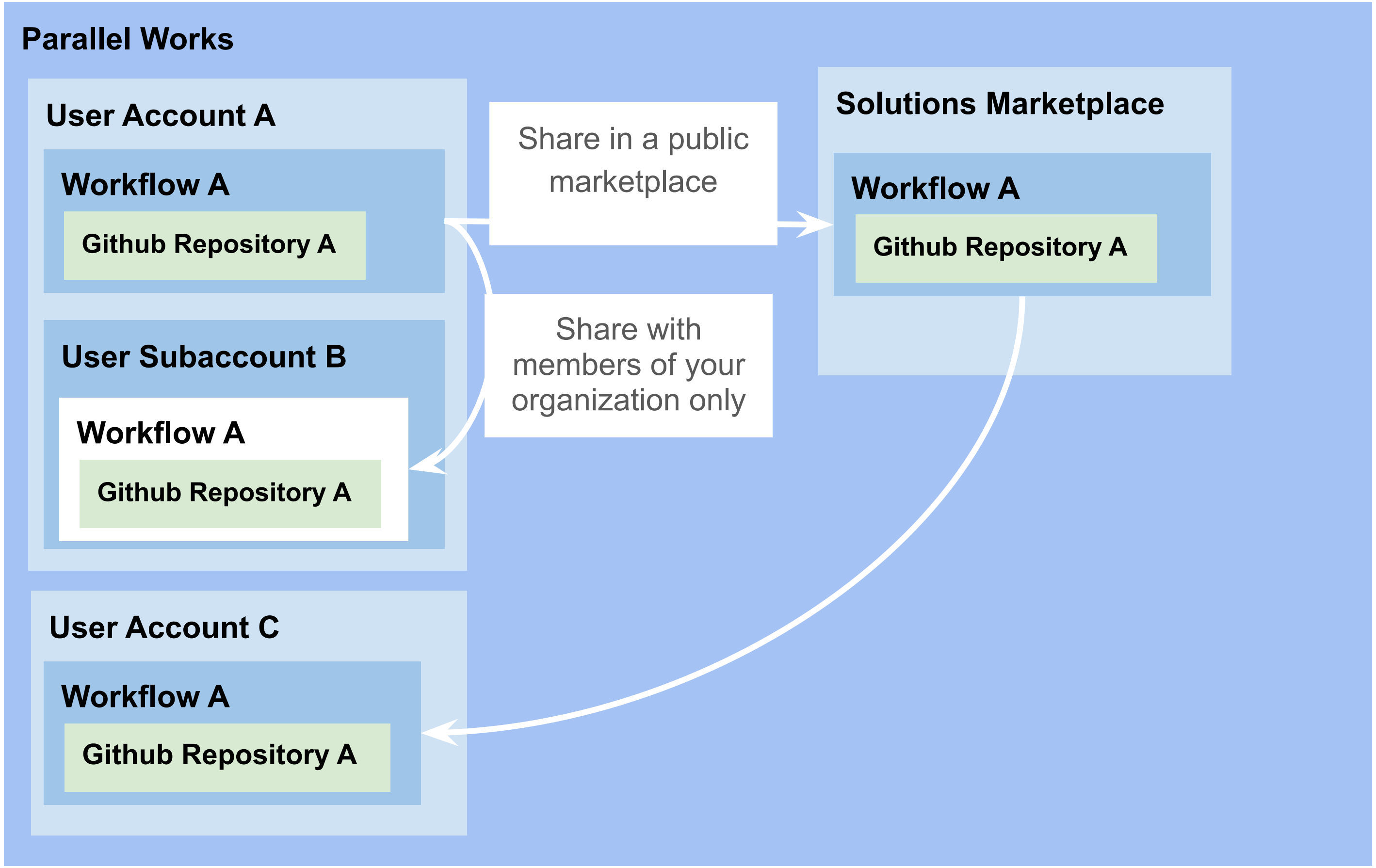
- User A needs read and write access

Users B and C use the workflow in production

- Need read access

Sharing Workflows in PW

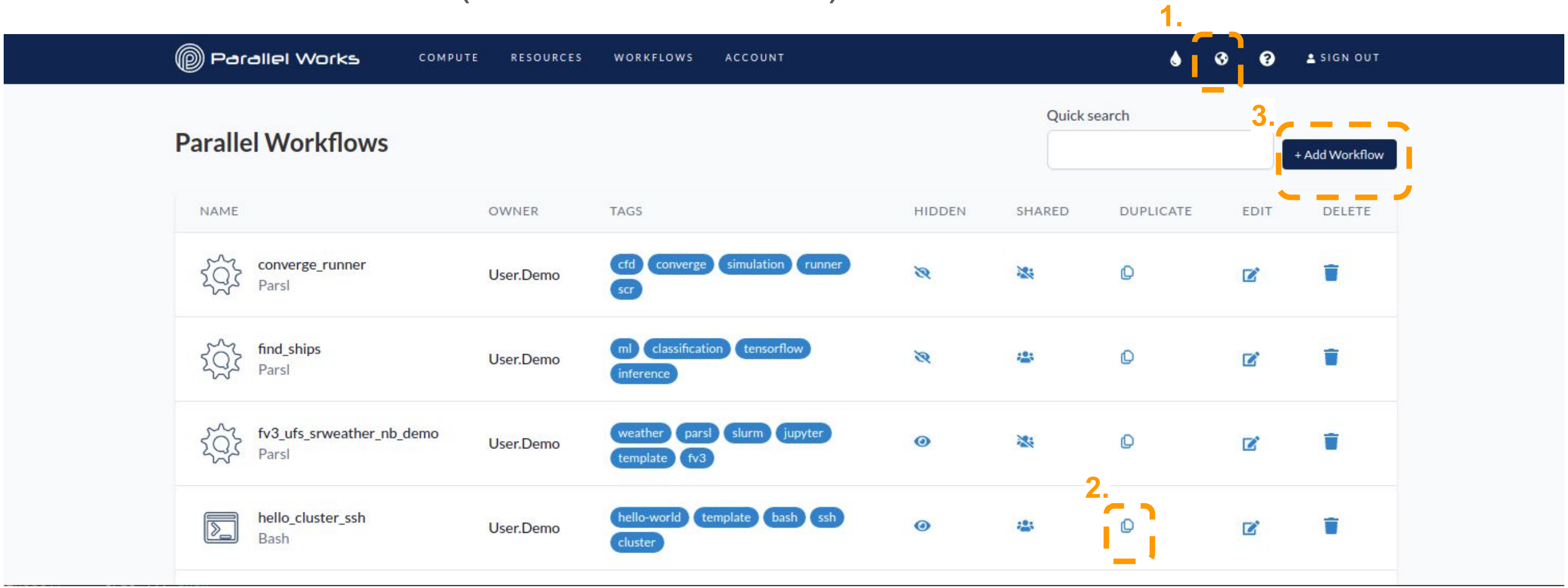
Can control read, write and admin access to your workflow in PW and/or use deploy keys in Github



Creating Workflows in PW

These are the options to create a workflow in PW:

1. Import a workflow from the solutions marketplace
2. Duplicate an existing workflow in your account
3. Add a new workflow (not recommended)



Parallel Workflows

Quick search

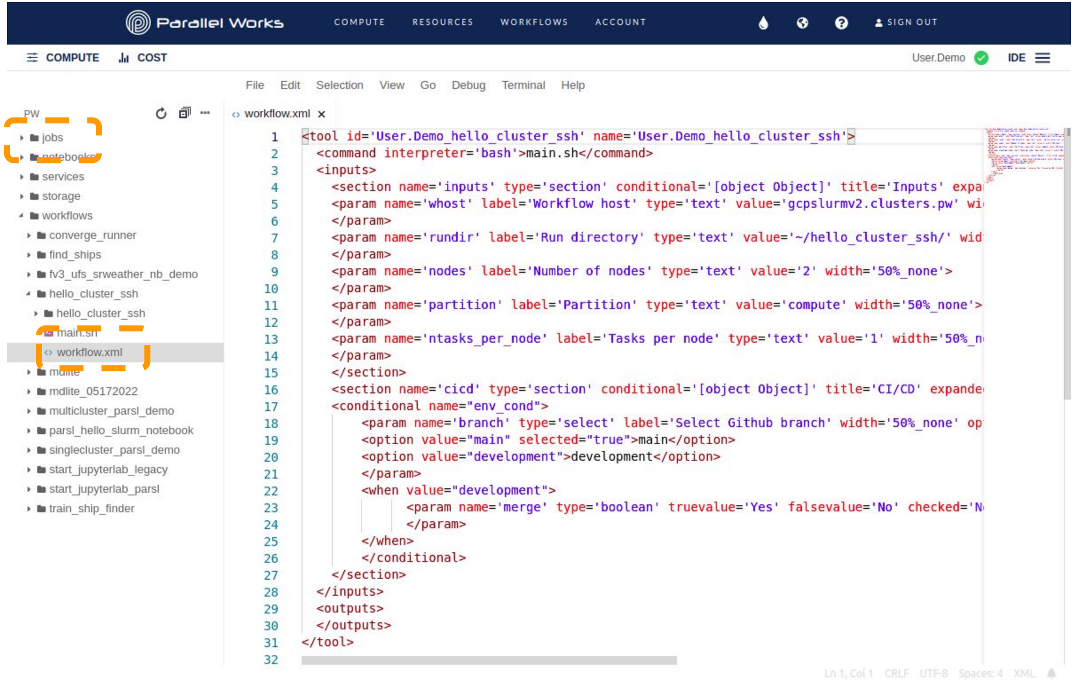
+ Add Workflow

NAME	OWNER	TAGS	HIDDEN	SHARED	DUPLICATE	EDIT	DELETE
converge_runner Parsl	User.Demo	cfd converge simulation runner scr					
find_ships Parsl	User.Demo	ml classification tensorflow inference					
fv3_ufs_srweather_nb_demo Parsl	User.Demo	weather parsl slurm jupyter template fv3					
hello_cluster_ssh Bash	User.Demo	hello-world template bash ssh cluster					

PW Jobs

When a workflow is executed a PW job is created

1. The workflow is copied to and executed in **/pw/jobs/job-number**
2. The workflow's input form, command and arguments are defined in the **/pw/workflows/workflow-name/workflow.xml** file. For example, the XML file below runs the command:



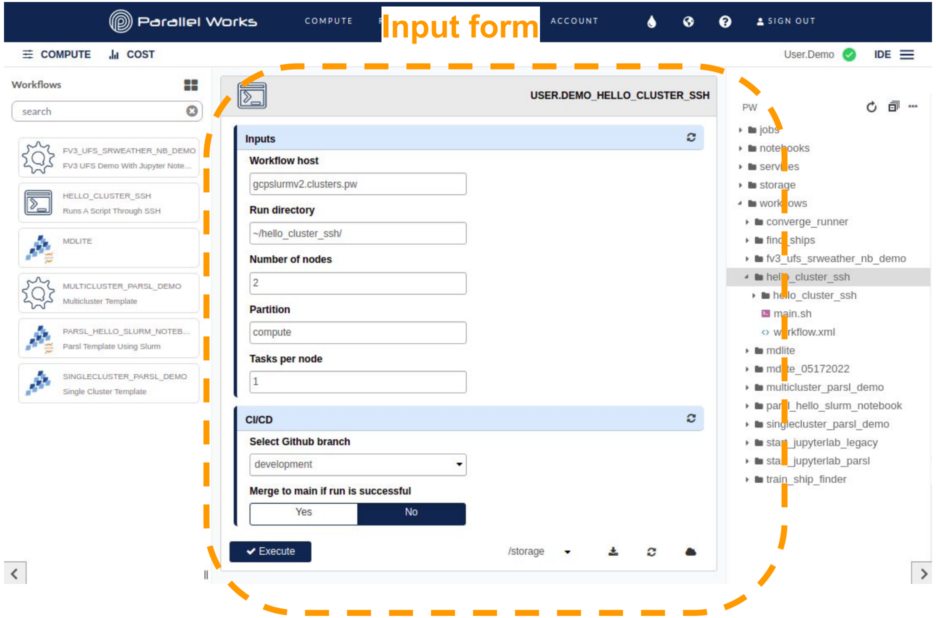
```xml
<tool id='User.Demo_hello_cluster_ssh' name='User.Demo_hello_cluster_ssh'>
  <command interpreter='bash'>main.sh</command>
  <inputs>
    <section name='inputs' type='section' conditional='[object Object]' title='Inputs' expa
    <param name='whost' label='Workflow host' type='text' value='gcpslurmv2.clusters.pw' wi
    </param>
    <param name='rundir' label='Run directory' type='text' value='~/hello_cluster_ssh/' wid
    </param>
    <param name='nodes' label='Number of nodes' type='text' value='2' width='50%_none'>
    </param>
    <param name='partition' label='Partition' type='text' value='compute' width='50%_none'>
    </param>
    <param name='ntasks_per_node' label='Tasks per node' type='text' value='1' width='50%_n
    </param>
    </section>
    <section name='cicd' type='section' conditional='[object Object]' title='CI/CD' expande
    <conditional name="env_cond">
        <param name='branch' type='select' label='Select Github branch' width='50%_none' op
        <option value="main" selected="true">main</option>
        <option value="development">development</option>
        </param>
        <when value="development">
              <param name='merge' type='boolean' truevalue='Yes' falsevalue='No' checked='N
              </param>
        </when>
        </conditional>
    </section>
  </inputs>
  <outputs>
  </outputs>
</tool>
```

```
bash main.sh \
    --whost gcpslurmv2.clusters.pw
\
    --rundir ~/hello_cluster_ssh/ \
    --nodes 2 \
    --partition compute \
    --ntasks_per_node 1 \
    --branch main \
```

Note that these parameter values are the default values for this workflow. Users may specify their own parameter values in the input form or the PW client (see next slide)

PW Jobs

Workflows can be executed from the input form (web UI) and using the PW client (automated)



Input form

USER.DEMO_HELLO_CLUSTER_SSH

Inputs

Workflow host: gcpslurmv2.clusters.pw

Run directory: ~/hello_cluster_ssh/

Number of nodes: 2

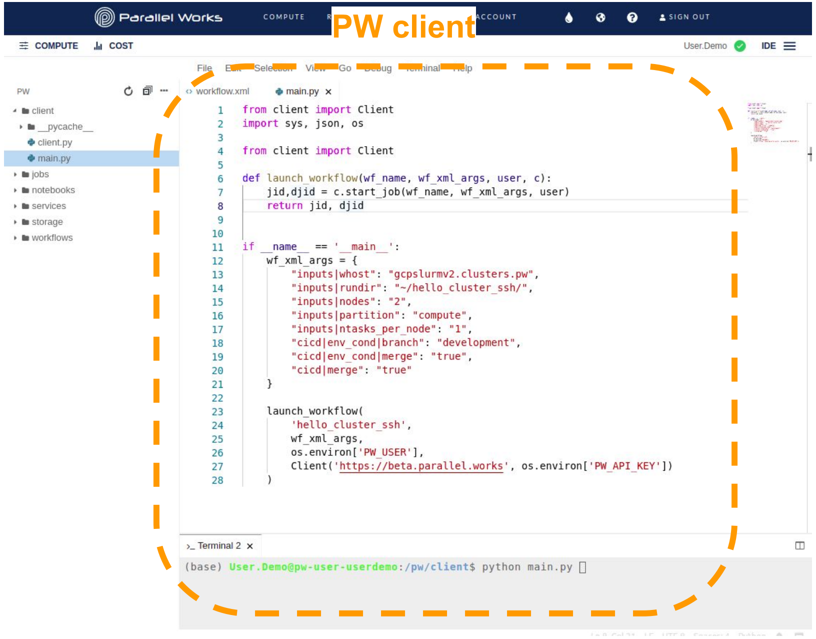
Partition: compute

Tasks per node: 1

CI/CD

Select Github branch: development

Merge to main if run is successful: Yes [✓ No]



PW client

```python
from client import Client
import sys, json, os

from client import Client

def launch_workflow(wf_name, wf_xml_args, user, c):
    jid,djid = c.start_job(wf_name, wf_xml_args, user)
    return jid, djid


if __name__ == '__main__':
    wf_xml_args = {
        "inputs|whost": "gcpslurmv2.clusters.pw",
        "inputs|rundir": "~/hello_cluster_ssh/",
        "inputs|nodes": "2",
        "inputs|partition": "compute",
        "inputs|ntasks_per_node": "1",
        "cicd|env_cond|branch": "development",
        "cicd|env_cond|merge": "true",
        "cicd|merge": "true"
    }

    launch_workflow(
        'hello_cluster_ssh',
        wf_xml_args,
        os.environ['PW_USER'],
        Client('https://beta.parallel.works', os.environ['PW_API_KEY'])
    )
```

```
(base) User.Demo@pw-user-userdemo:/pw/client$ python main.py
```

Github Deploy Keys

Use deploy keys to manage access of PW accounts to Github repositories. Follow these steps:

1. Create new ssh keys under **~/.ssh/org_name.repo_name.github.id_rsa** by running the following command in a terminal window of the PW IDE:

   ssh-keygen -t rsa

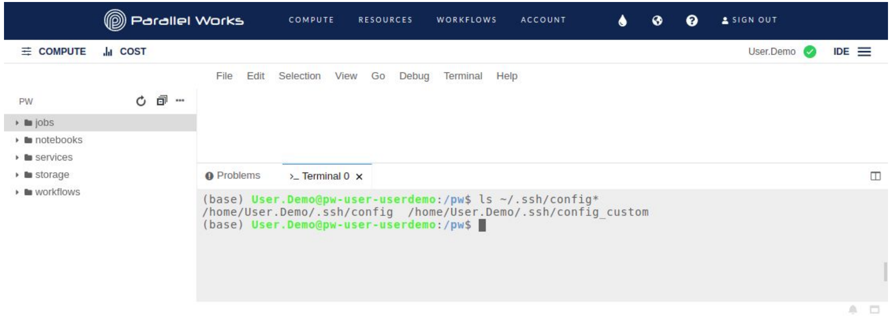
2. Create a new entry in the **~/.ssh/config** and **~/.ssh/config_custom** files:

   ```
   Host org_name-repo_name
     HostName github.com
     User git
     IdentityFile ~/.ssh/org_name-repo_name.github.id_rsa
   ```

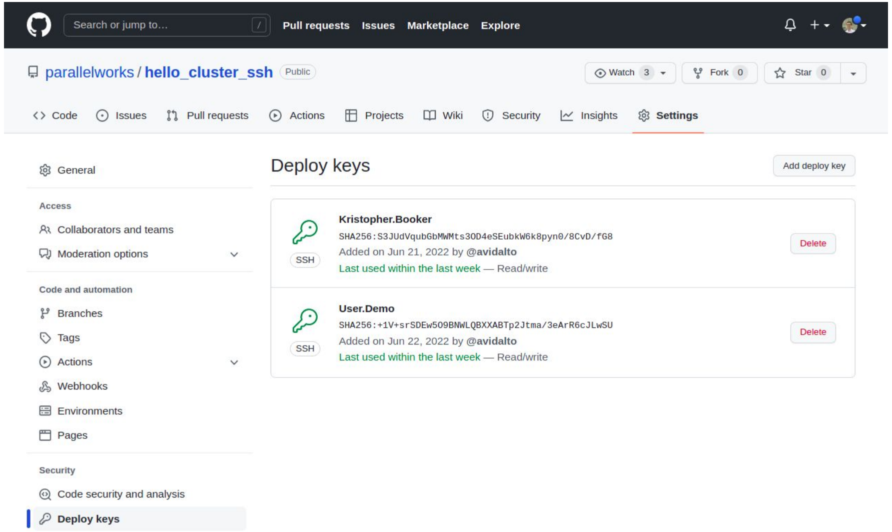
3. Add the public key to the deploy keys of the Github repository with read only or read and write permissions

4. Clone the repository with the command:

   git clone org_name-repo_name:org_name/repo_name.git



```
(base) User.Demo@pw-user-userdemo:/pw$ ls ~/.ssh/config*
/home/User.Demo/.ssh/config  /home/User.Demo/.ssh/config_custom
(base) User.Demo@pw-user-userdemo:/pw$
```



Deploy keys

Kristopher.Booker
SHA256:S3JUdVqubGbMWMts3OD4eSEubkW6k8pyn0/8CvD/fG8
Added on Jun 21, 2022 by @avidalto
Last used within the last week — Read/write
SSH
Delete

User.Demo
SHA256:+1V+srSDEw5O9BNWLQBXXABTp2Jtma/3eArR6cJLwSU
Added on Jun 22, 2022 by @avidalto
Last used within the last week — Read/write
SSH
Delete

Github Actions

Use Github actions to launch PW workflows using the PW API Client. An example action is provided in the repository: https://github.com/parallelworks/test-workflow-action

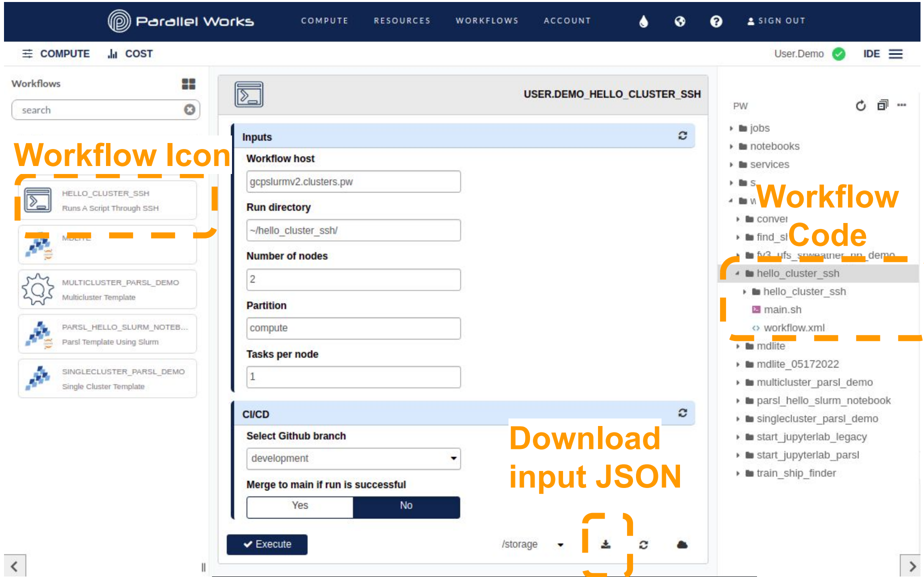
As an example, this action is used in the repository:

https://github.com/parallelworks/hello_cluster_ssh

(See .github/workflows/main.yaml file)

Where the workflow-parameters are downloaded from the input form of the hello_cluster_ssh in PW

The user's API key must be added to the secrets of the repository to be used by the PW API Client. This can be found in ACCOUNT > API Key or by printing the environment variable ${PW_API_KEY}



Workflow Icon

Workflow Code

Download input JSON

USER.DEMO_HELLO_CLUSTER_SSH

Inputs

Workflow host: gcpslurmv2.clusters.pw

Run directory: ~/hello_cluster_ssh/

Number of nodes: 2

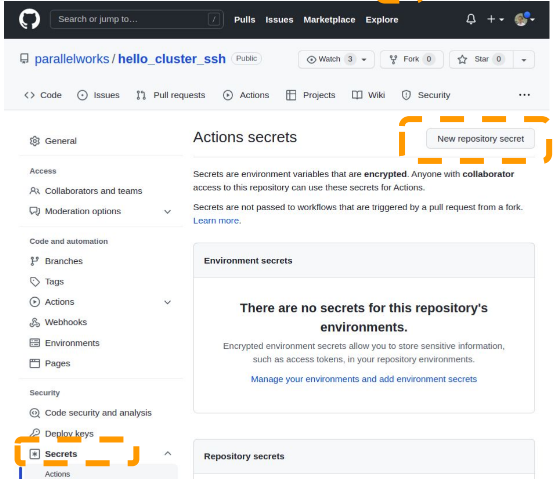
Partition: compute

Tasks per node: 1

CI/CD

Select Github branch: development

Merge to main if run is successful: Yes / No



parallelworks / hello_cluster_ssh

Actions secrets

New repository secret

Secrets are environment variables that are **encrypted**. Anyone with **collaborator** access to this repository can use these secrets for Actions.

Secrets are not passed to workflows that are triggered by a pull request from a fork. Learn more.

Environment secrets

There are no secrets for this repository's environments.

Encrypted environment secrets allow you to store sensitive information, such as access tokens, in your repository environments.

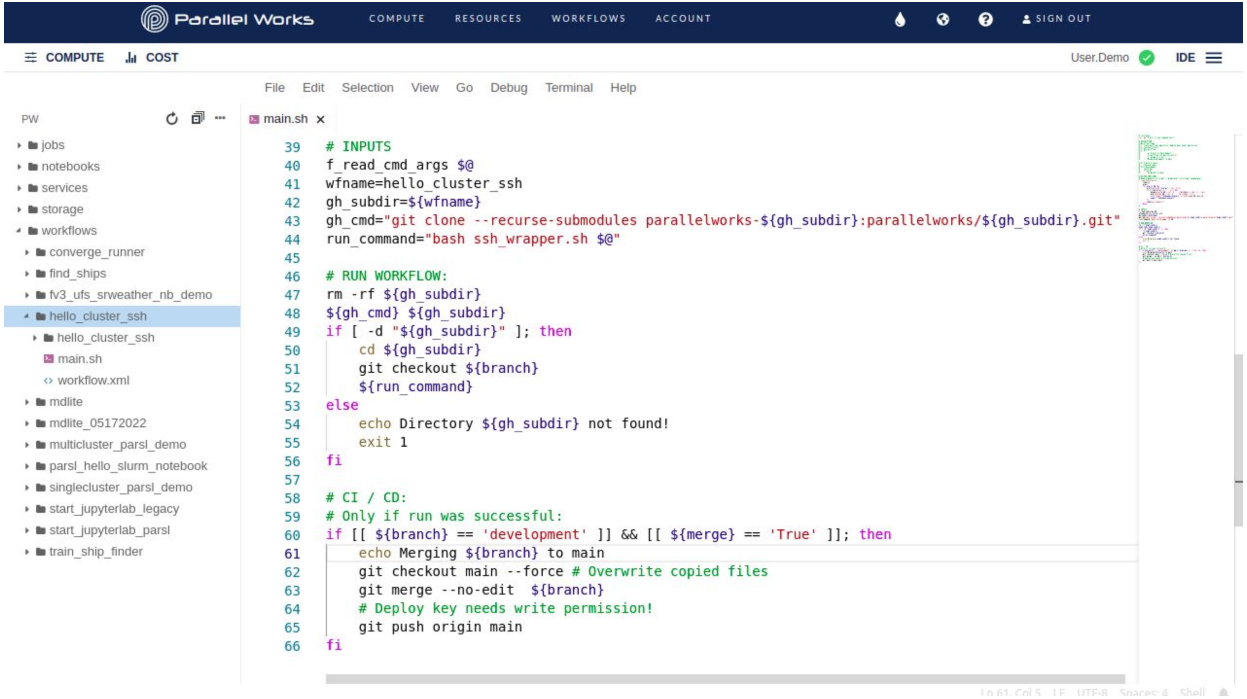
Manage your environments and add environment secrets

Repository secrets

Github Calls in PW Workflows

The hello_cluster_ssh workflow is an example of:

1. Cloning a github repository every time a workflow is executed (needs read access to the repository)
2. Merging two branches (development to main) if the workflow runs successfully (needs write access to the repository)



```
# INPUTS
f_read_cmd_args $@
wfname=hello_cluster_ssh
gh_subdir=${wfname}
gh_cmd="git clone --recurse-submodules parallelworks-${gh_subdir}:parallelworks/${gh_subdir}.git"
run_command="bash ssh_wrapper.sh $@"

# RUN WORKFLOW:
rm -rf ${gh_subdir}
${gh_cmd} ${gh_subdir}
if [ -d "${gh_subdir}" ]; then
    cd ${gh_subdir}
    git checkout ${branch}
    ${run_command}
else
    echo Directory ${gh_subdir} not found!
    exit 1
fi

# CI / CD:
# Only if run was successful:
if [[ ${branch} == 'development' ]] && [[ ${merge} == 'True' ]]; then
    echo Merging ${branch} to main
    git checkout main --force # Overwrite copied files
    git merge --no-edit  ${branch}
    # Deploy key needs write permission!
    git push origin main
fi
```

Tagging and Releasing a Repository Version

To add tag to a github repository run the following commands:

```
git tag -a -m "My new tag" vMAJOR.MINOR.PATCH
git push --follow-tags
```

Then go to Github releases (e.g.: https://github.com/parallelworks/hello_cluster_ssh/**releases**), select “draft a new release” and select your tag. This should trigger the action in the hello_cluster_ssh repository