Self-hosted Github runner

Requirement: Run P4CP (networking-recipe) Github action to validate commits by building against the ES2K P4SDE, without exposing SDE outside of Intel network.

Github offers free Github-hosted runners for CI jobs to be run as actions on repo commits.

A use-case for self-hosted runner would be to use OS which Github-hosted runners may not support (Gitub-hosted runners can only be Ubuntu Linux, Windows or MacOS types), or if we do not want to expose Intel IP to external resources.

In order to support an ES2K CI workflow, a self-hosted runner was created in lab.

Ubuntu 22.04 VMs were created on the following two systems in the P4CP lab.

P9-P10	JF3418-16B6-29839-P9	10.166.224.130	MEV Dayton Peak	p9-ubuntu-vm
	JF3418-16B6-28722-P10	10.166.231.240	CVL (Link Partner for P9)	p10-ubuntu-vm

Reason for choosing Ubuntu 22.04:

- It is one of the support OSes for P4 stack (P4SDE and P4CP)
- We already have artifacts from stratum-deps built by the standard Github-hosted runners. We can just download the tarball instead of rebuilding for Fedora or other Linux flavors
- Tofino target SDE only works on Ubuntu Linux
- P4SDE for ES2K target already has a CI from DevOps team that builds in the Ubuntu Linux environment
- Latest Ubuntu (24.04) is broken for P4CP, so we have pinned our support on all other CIs for Ubuntu 22.04

Step 1: Prepare the machines for self-hosted runners

If available, use a baremetal Ubuntu 22.04 system. If you need to install an Ubuntu 22.04 VM on one of the lab systems (say on a Fedora host OS system), see these instructions.

>> Steps to install Ubuntu VM on Fedora

Step 2: Install required tools, packages and libraries

2.1 Setup git using dt

Use https://1source.intel.com/onboard to setup git and authorize access to Intel repos

2.2 Clone SDE and P4CP repos

2.3 Install required packages

First install *python3-pip* and *distro*. Then, run the SDE's install_dep.py script, which installs all the required packages

```
apt install -y python3-pip
pip3 install distro

cd SDE/tools/setup
python3 install_dep.py
```

Install packages required by P4CP

```
apt install -y libatomic1 libnl-route-3-dev libssh-dev
cd P4CP; pip3 install -r requirements.txt
```

3. Add a user (gh-runner)

It is better to run the service as a separate user. Online resources suggest adding a non-login user, but on the systems above, a regular user has been created in order to allow login as gh-runner user and debug env problems.

```
adduser gh-runner
# (password is 'gh-runner')

sudo usermod -aG sudo gh-runner

# Verify user created
id gh-runner
groups gh-runner | grep sudo

# Verify gh-runner has sudo access
su - gh-runner
sudo whoami

# last command above should say 'root'
```

4. Add a self-hosted runner to the repository

- Go to github.com/ipdk-io/networking-recipe
- Go to Settings > Actions > Runners
- Click 'New self-hosted runner'
- Select Linux
- Execute all the commands on the provisioned Ubuntu machine
    - Select a unique label so jobs can be tagged and picked up (for es2k job, the label 'es2k-runner' was used)
    - First test by simply running the service in foreground mode (using the `./run.sh' command).
    - Verify it says "Listening to jobs" and the Github.com page should show the newly added runner as 'Idle'
- Configure to have the runner run as a service
    - Kill the './run.sh' process
    - Run following commands

```
sudo ./svc.sh install gh-runner

# The default service config file needs to be edited to add proxy settings. Edit to add the following details
(Environment and Restart fields). The final [Service] section should look like this:
sudo vi /etc/systemd/system/actions.runner.ipdk-io-networking-recipe.p9-ubuntu-vm.service

        [Service]
        ExecStart=/home/github-runner/actions-runner/runsvc.sh
        WorkingDirectory=/home/github-runner/actions-runner
        User=github-runner
        Group=github-runner
        KillMode=process
        KillSignal=SIGTERM
        TimeoutStopSec=5min
        Environment="PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
        Environment="http_proxy=http://proxy-dmz.intel.com:911"
        Environment="https_proxy=http://proxy-dmz.intel.com:912"
        Environment="NO_PROXY=localhost,127.0.0.1"      # Exclude local addresses from proxy
        Restart=always


sudo systemctl daemon-reload
sudo ./svc.sh start

# give it a few seconds and check status.
sudo ./svc.sh status

# This should say "connected to Github.com" if successful (and the Github.com Runners page should again say
'idle'
```

5. Compile P4SDE and stage it

To confirm that we have the Linux environment configured and all libs are installed, compile P4SDE (and other components once) to confirm that everything compiles ok

In case of P4SDE, the following had to be added/updated

```
find /usr/lib -name "libpython3*.so*"
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/x86_64-linux-gnu/
sudo ln -s /usr/lib/x86_64-linux-gnu/libpython3.10.so /usr/lib/x86_64-linux-gnu/libpython3.so
```

Once compiled, move the P4SDE to /opt directory and let gh-runner be the owner

This may not be necessary in future, if we can pull a tarball from existing CI job artifact.

```
mkdir -p /opt/p4dev/es2k-sde
chmod 777 /opt/p4dev/es2k-sde/
cp -pr SDE/install/* /opt/p4dev/es2k-sde/
chown -R gh-runner:gh-runner /opt/p4dev/es2k-sde/
```

6. Generate Organization-Wide Fine-Grained Token for accessing Intel repos

6.1 Create a fine-grained PAT

- Go to your Github personal page
- Go to Settings > Developer Settings > Personal Access Tokens > Fine-grained tokens (https://github.com/settings/personal-access-tokens)
- Generate new token and select IPDK organization as resource owner
- Grant permissions such as 'Contents (read-only or read/write)'
- Save token securely – you will not be able to view it again later

6.2 Store the token as an organization secret

- Go to IPDK org page
- Navigate to Settings > Secrets & Variables (under Security) > Actions (https://github.com/organizations/ipdk-io/settings/secrets/actions)
- Create New Organization Secret and use the fine-grained PAT created in previous step. Save it as 'ORG_PAT' (this will be used as key to refer to secret in workflow files)

7. Update Workflow file

Update the workflow: https://github.com/ipdk-io/networking-recipe/pull/754

Of things to note are the "*self-hosted*" type of runner and "*es2k-runner*" label.

This will invoke the job on our newly configured self-hosted runner

Access to repos uses the *'secrets.ORG_PAT'*, which comes from the Github Action Secrets, configured in previous step.