Building an Ephemeral Docker Image and Container for Terraform

Introduction

This Troubleshooting Guide (TSG) walks you through the process of setting up an ephemeral Docker container in AWS CloudShell to run Terraform. Follow these steps carefully to avoid common build and execution errors.

Step 1: Ensure Docker is Available in AWS CloudShell

AWS CloudShell does not always have Docker installed by default. Run the following command to check:

```
docker --version
```

If Docker is not installed, you may need to use an alternative environment like an EC2 instance.

- In my experience it's on there already...

Step 2: Create the Dockerfile

Use a text editor (e.g., nano or VS Code) to create a Dockerfile with the following content:

```
FROM amazon/aws-cli:2.11.4
RUN yum update -y && yum install -y git unzip
RUN git clone https://github.com/tfutils/tfenv.git /root/.tfenv && \
    ln -s /root/.tfenv/bin/* /usr/local/bin && \
    tfenv install latest && \
    tfenv use latest
WORKDIR /work
ENTRYPOINT ["/bin/bash"]
```

Save this file as '(your filename).dockerfile'. You will use this to build your image.

Step 3: Build the Docker Image

Navigate to the directory where your Dockerfile is located and run:

```
docker build -t terraform_docker -f (your filename).dockerfile .
```

This will create a reusable Docker image named 'terraform_docker'. If you see an error like:

```
ERROR: failed to solve: failed to read dockerfile
```

Make sure you are in the correct directory and that the file exists.

Step 4: Run the Container

Once the image is built, start a new container with:

Building an Ephemeral Docker Image and Container for Terraform

```
docker run -it --rm -v ${PWD}:/work terraform_docker
```

This command does the following:
- `-it`: Runs in interactive mode with a terminal.
- `--rm`: Deletes the container when it stops.
- `-v ${PWD}:/work`: Mounts the current directory to '/work' inside the container.

If CloudShell times out, the container will be destroyed, but the image remains available for reuse.

Step 5: Verify Terraform is Installed

Inside the running container, check the Terraform version:

```
terraform --version
```

You should see output confirming that Terraform is installed. If it is missing, ensure that 'tfenv' installed the latest version.

Troubleshooting Common Issues

1. **Dockerfile Not Found Error**
   - Ensure you are in the correct directory and using `-f (your filename).dockerfile`.

2. **Permission Denied When Running Docker**
   - If CloudShell does not allow Docker, use an EC2 instance instead.

3. **Terraform Not Found Inside Container**
   - Run `tfenv install latest` and `tfenv use latest` inside the container.

4. **Session Timeout & Container Deletion**
   - Since `--rm` removes the container after exit, restart it by running `docker run` again.

5. **Unable to Mount Files into Container**
   - Ensure you are running Docker from the directory containing your Terraform files.

Conclusion

You have successfully built an ephemeral Docker image for Terraform in AWS CloudShell. This setup allows you to manage AWS resources efficiently without installing Terraform directly on CloudShell. If you encounter issues, refer to the troubleshooting section or rebuild the image.

Crafted and Edited:

Joseph Luster