Figure 10-3. Atlantis can automatically add the output of the `terraform plan` command as a comment on your pull requests.

Terraform Cloud and Terraform Enterprise, HashiCorp’s paid tools, both support running `plan` automatically on pull requests as well.

Merge and Release

After your team members have had a chance to review the code changes and `plan` output and all the tests have passed, you can merge your changes into the `main` branch and release the code. Similar to application code, you can use Git tags to create a versioned release:

```
$ git tag -a "v0.0.6" -m "Updated hello-world-example text"
$ git push --follow-tags
```

Whereas with application code, you often have a separate artifact to deploy, such as a Docker image or VM image, since Terraform natively supports downloading code from Git, the repository at a specific tag *is* the immutable, versioned artifact you will be deploying.

Deploy

Now that you have an immutable, versioned artifact, it’s time to deploy it. Here are a few of the key considerations for deploying Terraform code:

- Deployment tooling
- Deployment strategies
- Deployment server
- Promote artifacts across environments

Deployment tooling

When deploying Terraform code, Terraform itself is the main tool that you use. However, there are a few other tools that you might find useful: