Chosen folder structure

My folder structure has good maintainability and reusability because it keeps the different component/infrastructure configurations separate in different modules. This helps to easily track the code and keep it clean (less spaghetti code).

I feel like having one infrastructure template (modules folder) with 3 different .tfvars files for dev, staging, prod makes it so much easier to work and manage the code and configure without duplicating (or tripling) the code.

Scaling is also easy by just adding a new module folder. This will not affect the other modules and I can just add the new module in main.tf with the variables for all .tfvars files and that's it.

This structure is good for implementing CI/CD workflows because I can simply set up workflows with the specific .tfvars files that I have inside modules for each specific environment.

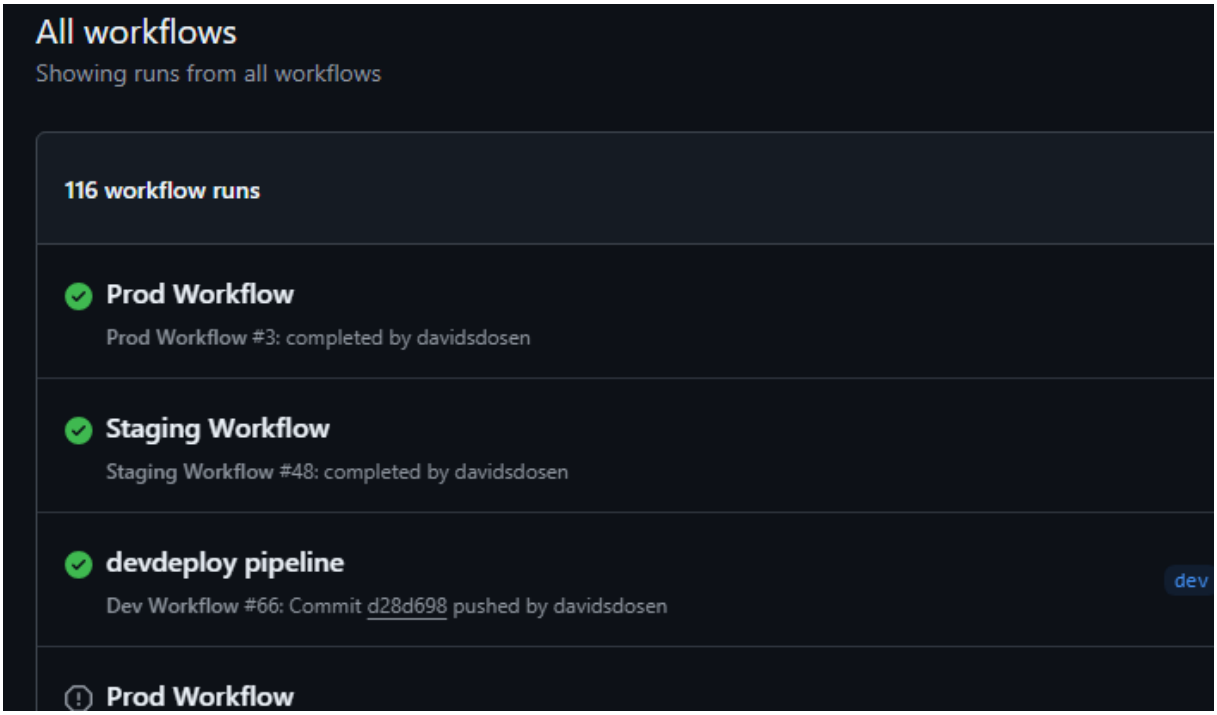
Challenges during development

- Spent lots of time fixing security issues so my code would pass the tfsec workflow check in dev workflow. Things like having an open port * when instead you should have a smaller IP range that can access my network, adding a minimum TLS version to my storage account along with many other small things to fix/implement.
- I was confused at first because of how nested and long the definition of the CI/CD pipeline task on the blackboard page is so I had to spend some time figuring out how to set up the the corresponding steps for the github actions pipeline inside the .yml files
- Figuring out how to do the automatic pull & merge was challenging in the .yml code but also because the GITHUB_TOKEN did not have read&write permissions by default and github actions automatic pull request approvals were are also turned off by default so I spent some time figuring that out and which setting that was. Also understanding how to properly code automatic merging is something I'm still working on. See next below.
- Issues which I think I managed to solve just in time:
    - When automatically merging at the end of a workflow, for example dev to staging, I got lots of merge conflicts at first because it was challenging for me to code it so it would force merge instead of just making a normal PR where I manually need to solve conflicts. I think I finally managed to make it work properly. This could be because of my lack of experience in working with git, especially related to merging.

- Sometimes the next workflow started before the previous one finished when the merge to the next branch from the previous workflow failed, it still marked the failed workflow where a merge would fail as completed and would start a run on an older version of the code in the next branch which was not intended.

```yaml
on:
  workflow_run:
    workflows: ["Dev Workflow"]
    types:
      - completed
    branches:
      - dev
```

- It took me a whole 116 workflow runs before I managed to finally run the whole pipeline from start to finish without failure, deploying dev and staging and then prod after manual approval.



Potential improvements or optimizations

- Maybe adding readme in each module to future proof the code base but at the same time I already commented the code
- If I want to make this just like a professional development environment, I can add extra protections for example nobody can push directly to the prod branch, so everything is forced to go through the whole workflow process chronologically.
- Previous assignment general feedback talks about having resources abstracted (separating resources into different modules). This task instructs me to make a load balancer in front of the web application, so I just included it inside another module. I don't know if this is correct so if it's wrong, a quick improvement would be making load balancer a separate module.
- Probably somewhere in the code I could have made it more flexible.