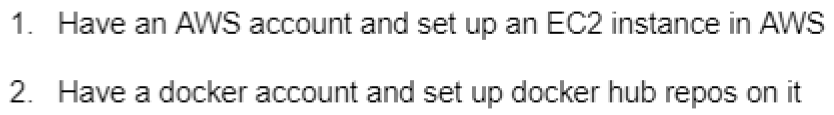
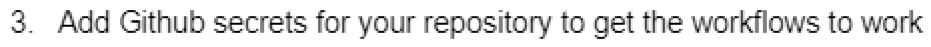
<u>CI/CD Workflow</u>

Previous BucStop group already had a functional workflow for automatic deployment, though it would take a few more steps compared to manual deployment.



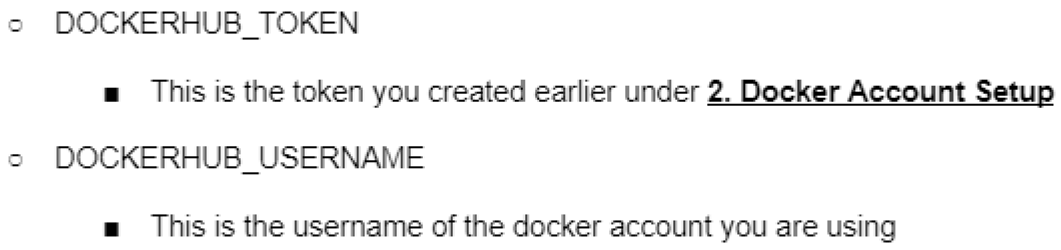
1. Have an AWS account and set up an EC2 instance in AWS
2. Have a docker account and set up docker hub repos on it

These two steps are already part of the deployment process, so they shouldn't be a challenge.

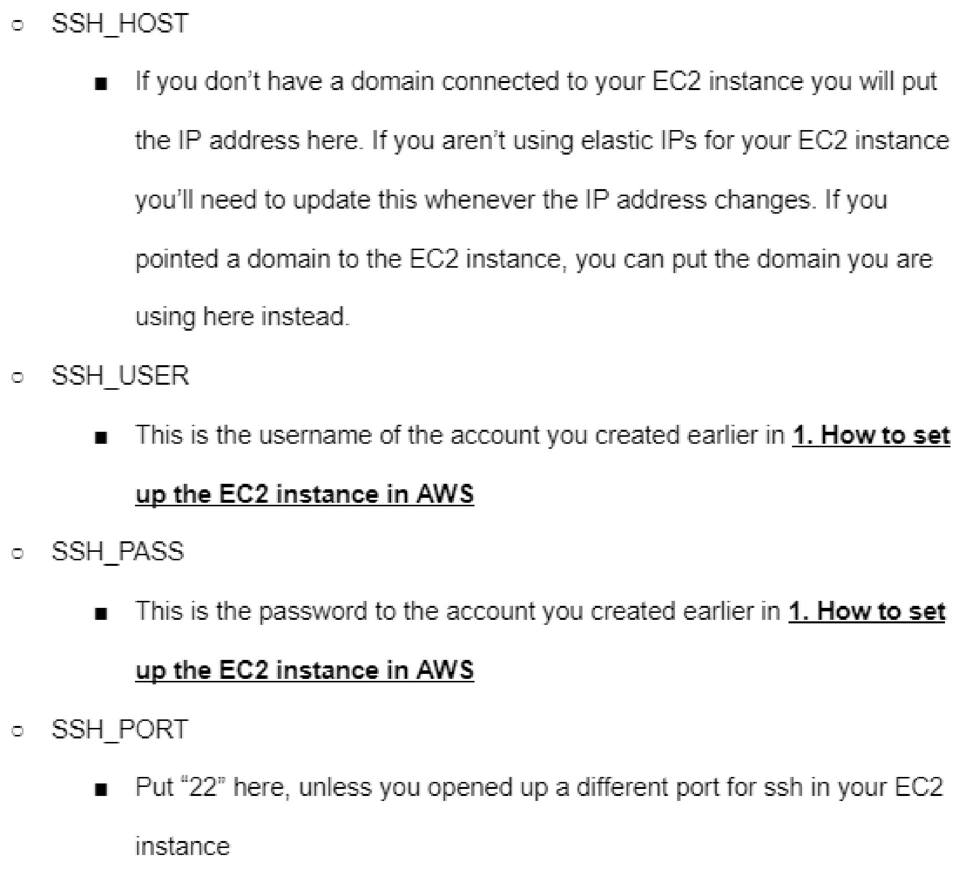


3. Add Github secrets for your repository to get the workflows to work

This is the main extra step for setting up automated deployment. The "How to Deploy" document also shows how to enable HTTPS traffic and attach an elastic IP to the AWS instance, but I don't think it is fully necessary for automated deployment.



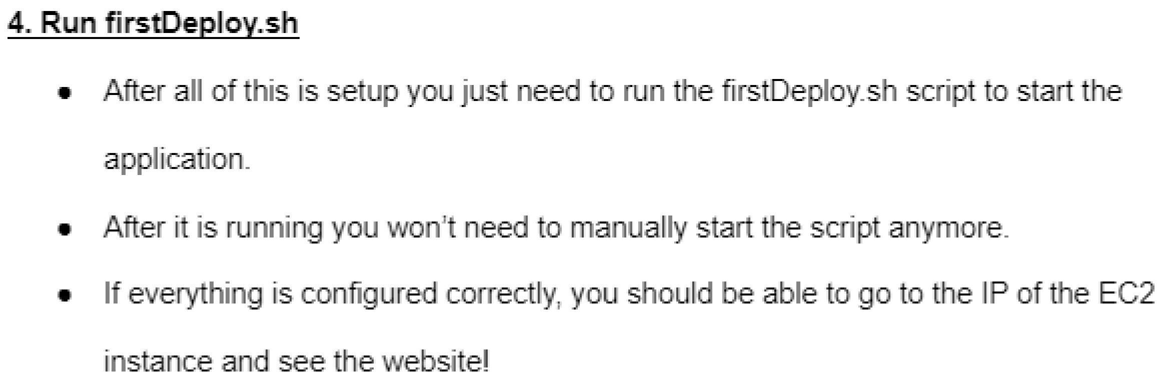
- DOCKERHUB_TOKEN
  - This is the token you created earlier under **<u>2. Docker Account Setup</u>**
- DOCKERHUB_USERNAME
  - This is the username of the docker account you are using



- SSH_HOST
  - If you don't have a domain connected to your EC2 instance you will put the IP address here. If you aren't using elastic IPs for your EC2 instance you'll need to update this whenever the IP address changes. If you pointed a domain to the EC2 instance, you can put the domain you are using here instead.
- SSH_USER
  - This is the username of the account you created earlier in **1. How to set up the EC2 instance in AWS**
- SSH_PASS
  - This is the password to the account you created earlier in **1. How to set up the EC2 instance in AWS**
- SSH_PORT
  - Put "22" here, unless you opened up a different port for ssh in your EC2 instance

These are the GitHub secrets the document show. The problem here is that the secrets are associated with a now deprecated repo. The main repo that the group seemed to be using was the API Gateway repo, which has is not mentioned in this document.

There is also no mention of the 3 separate game repos, but I believe the workflow would be similar as far as setting up GitHub secrets.



4. Run firstDeploy.sh

- After all of this is setup you just need to run the firstDeploy.sh script to start the application.
- After it is running you won't need to manually start the script anymore.
- If everything is configured correctly, you should be able to go to the IP of the EC2 instance and see the website!

This is the final step of the “How to Deploy” document. I’m not 100% sure what it means by “you won’t need to manually start the script anymore.” It may mean after a push is made on GitHub, but I am not completely sure.

This video from the previous team shows their recommendations for DevOps and the scrum processes, part of which includes automation and CI/CD. They discuss the importance of not focusing too heavily on automation, stating that it should be helpful, not a hindrance.

https://www.youtube.com/watch?v=dkfyNcr4gBA&list=PLxsGO-QGipWmVzxFkVbA-o6BUW5eRdk3H&index=4&t=225s

They mention some tools that they use, like Zapier integrated with Discord, which can be found here:

https://zapier.com/apps/chatbot/integrations/discord

They also mention Jenkins, a tool that helps to automate deployments by using Discord webhooks to check the status of a build. It can be found here:

https://plugins.jenkins.io/discord-notifier/

There is also GitHub actions (which I am not sure if the previous team used) that can be used to assist with CI/CD. It uses “Workflows” that are defined by a yaml file triggered by some event in a repository. More info can be found here:

https://docs.github.com/en/actions/about-github-actions/understanding-github-actions