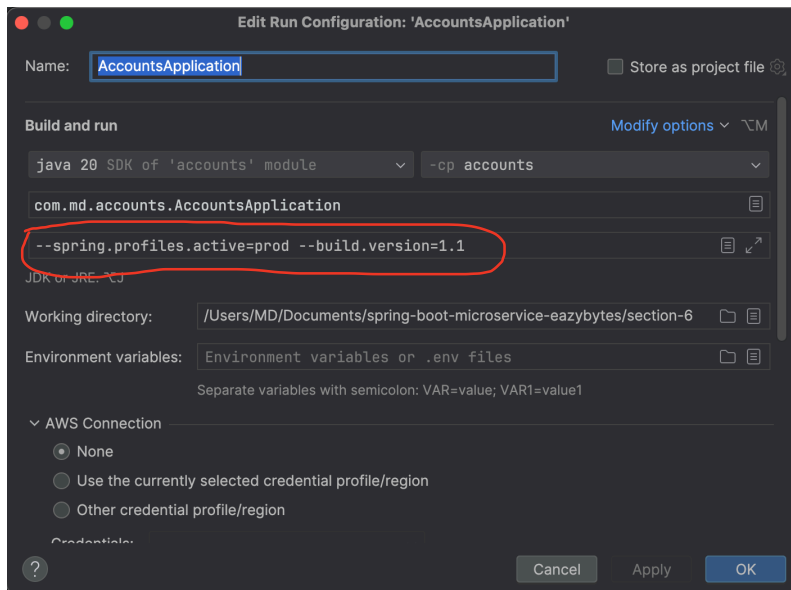
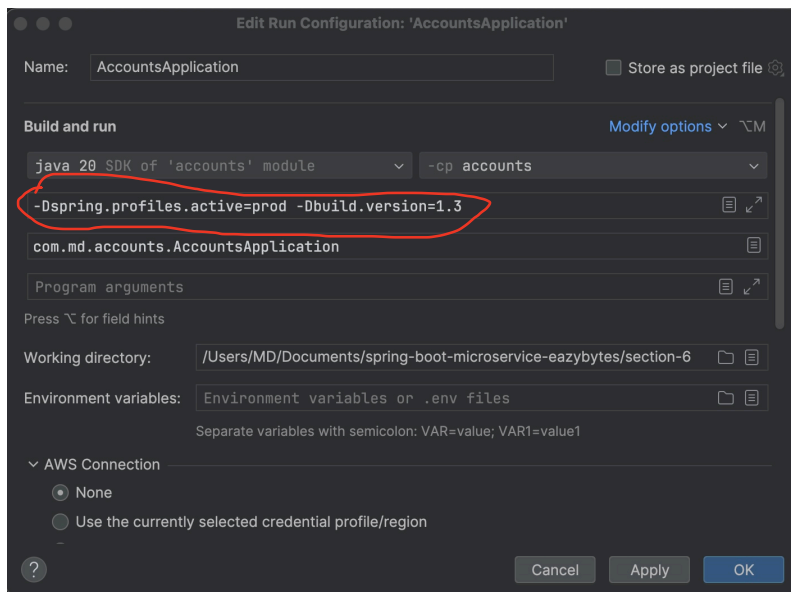


## There are three ways to configure the properties inside microservices,

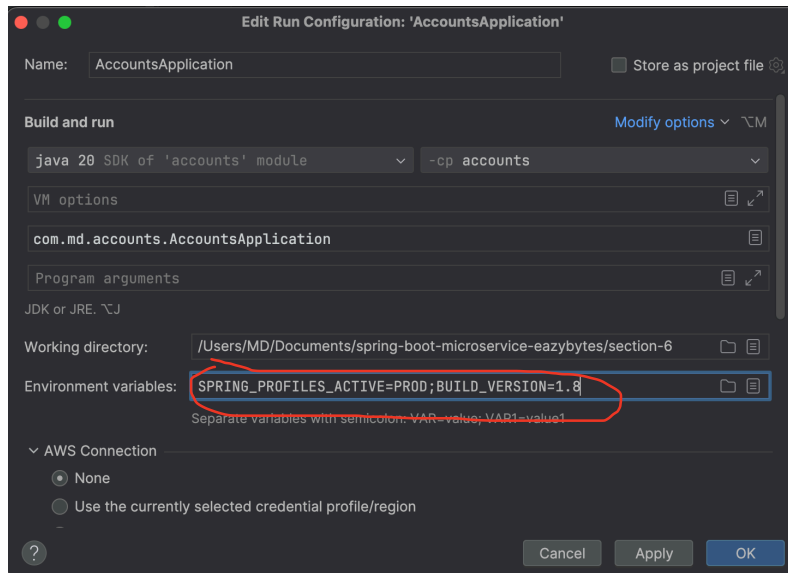
- one is hardcoding within application and later creating a separate class (like DTO) to access those properties and later create controller to showcase them, if needed.
- Spring profiles also an feasible solution but it has it's own drawbacks, though one can use external configurations using command line, JVM system and Environment variables to dynamically active profiles at the runtime. for more info on benefits look at the notes.



Using Command line arguments, we can even override the properties defined inside .yaml files



First we need to enable "Add VM Options" then use the command shown in image.



using Environment variables.


Refer to official documents of spring cloud config where spring team has demonstrated how to load config server when working with cloud provider like AWS and Google Cloud.

### Spring cloud config-server

- added Github as the central repository (refactored application.yml file inside configserver and other microservices)
- Modified records class (for eg: AccountContactDto) to normal java class as we aren't reading any configurations locally within project
- we implemented the functionality through which our microservices were able to fetch the updated configurations just like configserver application. For more info on how to implement it, look at the notes. please note that these option is feasible only if we have one or two instance of the application if we have more than that then we need a mechanism that automate this process efficiently for refreshing the configurations instead of manually integrating each application instance.
- To over come the above challenge, we are leveraging spring cloud bus, which uses brokers like Kafka, RabbitMQ. Follow video #85 for installations.
- later added "spring-cloud-monitor" dependency for monitoring configurations updates, also added webhooks to Github, this helps us to skip the manual step of hitting spring cloud bus API and auto fetches the updated configurations to all our microservices (please note that this requires to run RabbitMQ locally in background and we are still using "localhost" label within webhook (more for info: visit the video #86))
- please note this website where we have created a permanent webhook url: <https://console.hookdeck.com/> , once navigated open terminal and use below commands as shown images and later copy the url and go to webhooks option on Github's configserver setting section and paste this webhook url and we are good to go.

```
MD@Mauliks-MacBook ~ % hookdeck login --cli-key 4ezh70rmuonjkltpnyk10ummv3vqmqr7vv8o8sfw0n7rbfhk
> Done! The Hookdeck CLI is configured with your console Sandbox
MD@Mauliks-MacBook ~ % hookdeck listen [port] Source
zsh: no matches found: [port]
MD@Mauliks-MacBook ~ % hookdeck listen 8071 Source
▶ Not connected with any account. Creating a guest account...
? What path should the webhooks be forwarded to (ie: /webhooks)? /monitor
? What's your connection label (ie: My API)? localhost
```

#### Dashboard

 Console URL: <https://api.hookdeck.com/signin/guest?token=57nptmw9pzej25jma4ph06vd8dhahx7msv6pd7dillfujpzdaa>  
Sign up in the Console to make your webhook URL permanent.

#### Source Source

 Webhook URL: <https://hkdk.events/i2o9u6u68xxz8o>

#### Connections

localhost forwarding to /monitor

> Ready! (^C to quit)

█