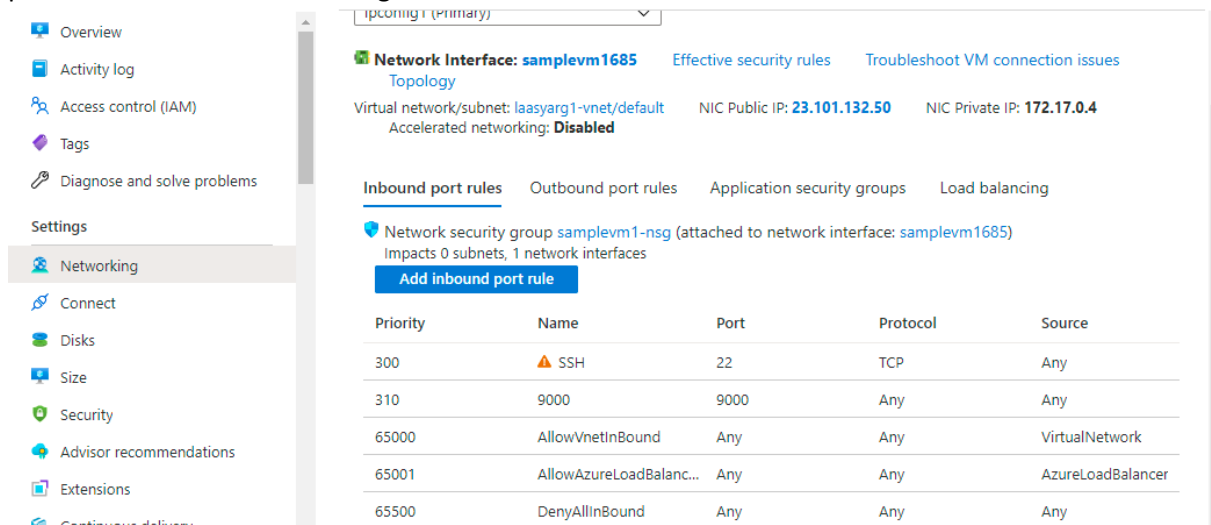


INTEGRATING SONARQUBE WITH AZURE DEVOPS FOR CODE INSPECTION

1. Login to the VM created.
2. Install docker using: `$sudo apt install docker.io`
3. Now pull sonarqube using docker: `$sudo docker pull sonarqube:latest`
4. Run the container: `sudo docker container run -d --name sonarqube -p 9000:9000 sonarqube:latest`

```
azureuser@samplevm1:~$ docker container run -d --name sonarqube -p 9000:9000 sonarqube:latest
docker: Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/create?name=sonarqube: dial unix /var/run/docker.sock: connect: permission denied.
See 'docker run --help'.
azureuser@samplevm1:~$ sudo docker container run -d --name sonarqube -p 9000:9000 sonarqube:latest
cf437f31e827f46668aa195840c616c92f984e402a063f48054cf8f05e4b9ffa
azureuser@samplevm1:~$ docker container ps
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/json: dial unix /var/run/docker.sock: connect: permission denied
azureuser@samplevm1:~$ sudo docker container ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
cf437f31e827   sonarqube:latest  "bin/run.sh bin/sona..."  27 seconds ago  Up 26 seconds  0.0.0.0:9000->9000/tcp   sonarqube
```

5. open browser and type <pub_ip_of_vm>:9000. Note: Make sure to add an inbound rule for port 9000 in the VM networking section.

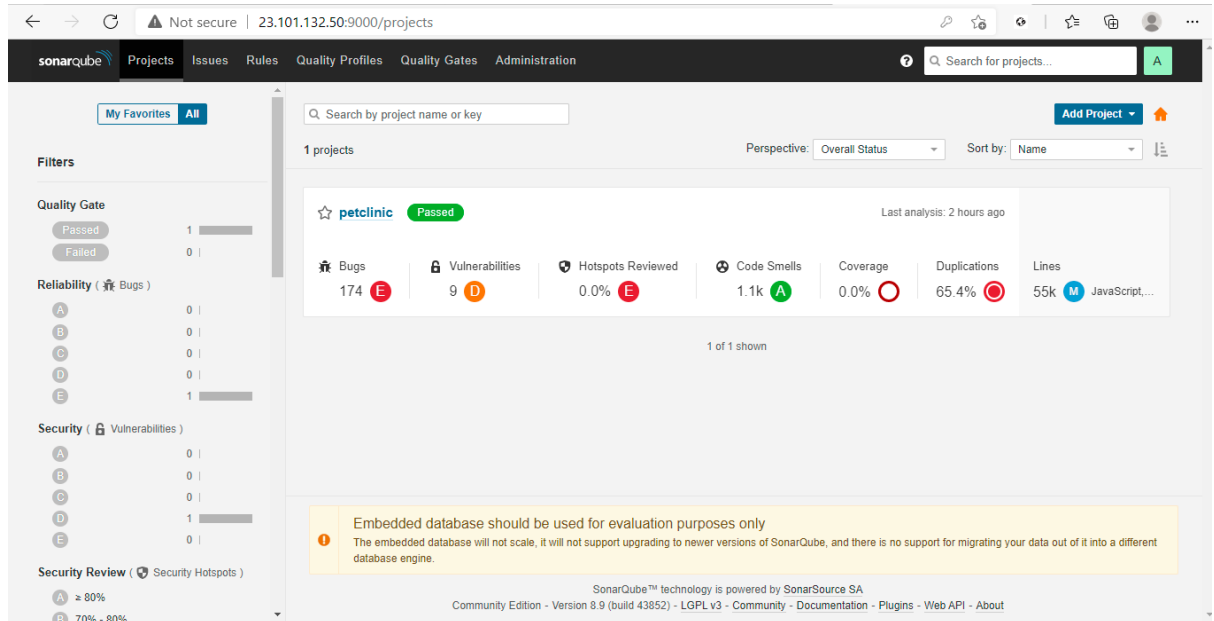


The screenshot shows the Azure portal interface for a VM named 'samplevm1685'. The 'Networking' section is expanded, and the 'Inbound port rules' tab is selected. The 'Network security group' is 'samplevm1-nsg'. A table of inbound port rules is displayed, with the rule for port 9000 highlighted.

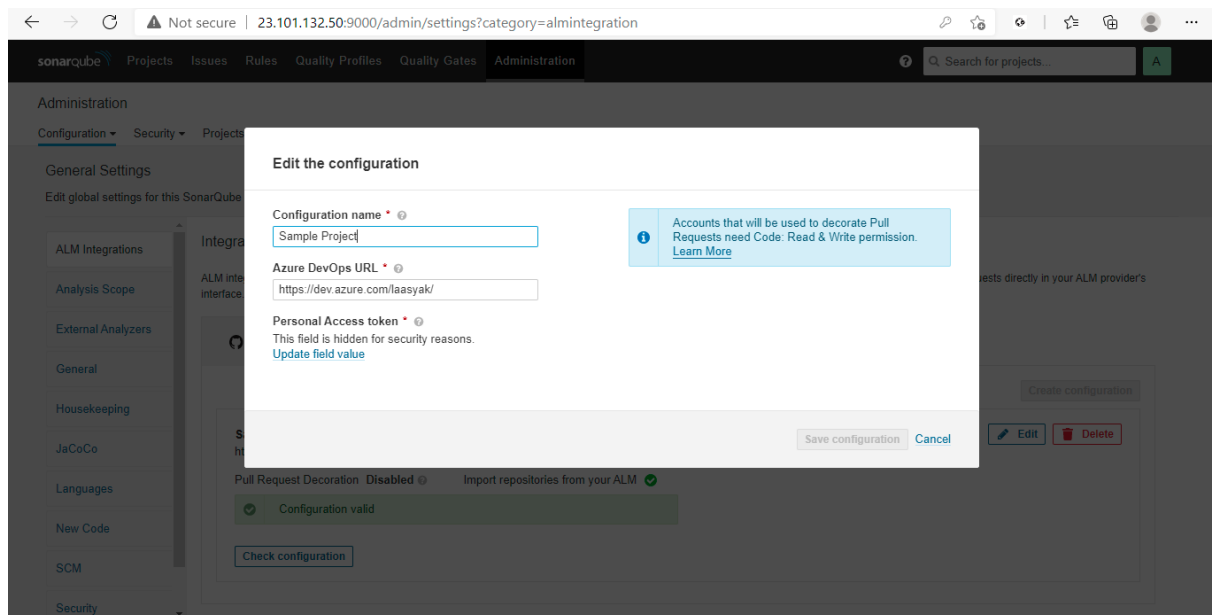
Priority	Name	Port	Protocol	Source
300	SSH	22	TCP	Any
310	9000	9000	Any	Any
65000	AllowVnetInBound	Any	Any	VirtualNetwork
65001	AllowAzureLoadBalanc...	Any	Any	AzureLoadBalancer
65500	DenyAllInBound	Any	Any	Any

6. Now you will be able to see the login page of the sonarqube. Username and password are admin by default. You will then be prompted to change the password.

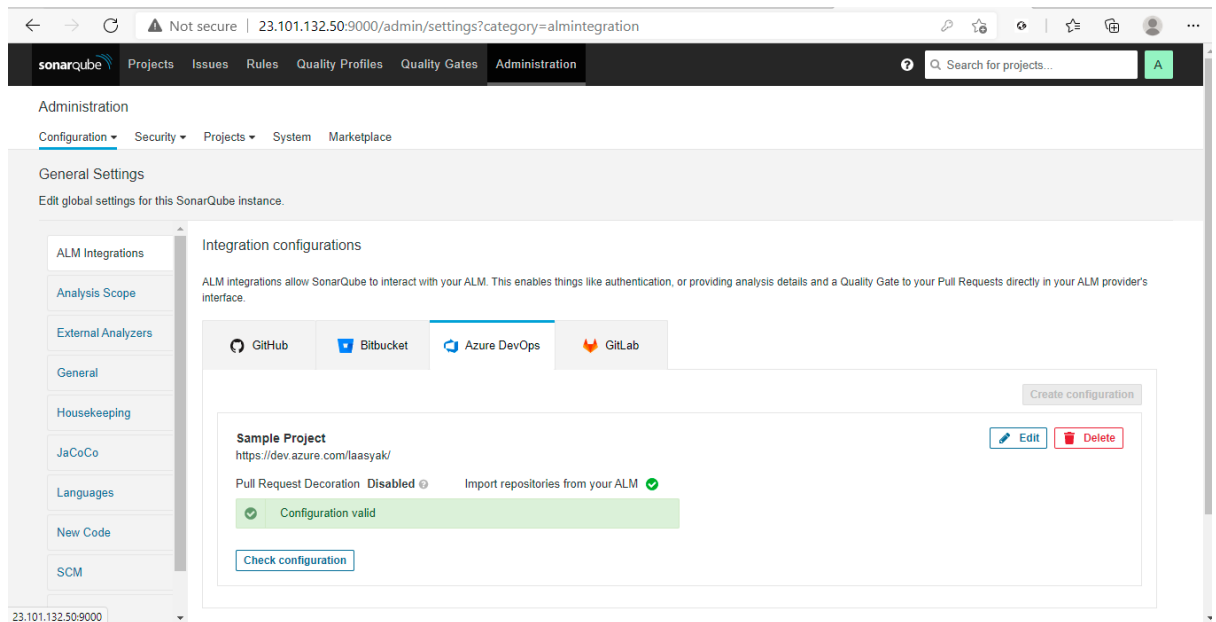
7. Once that is done, you will be able to see the dashboard with no attached projects.



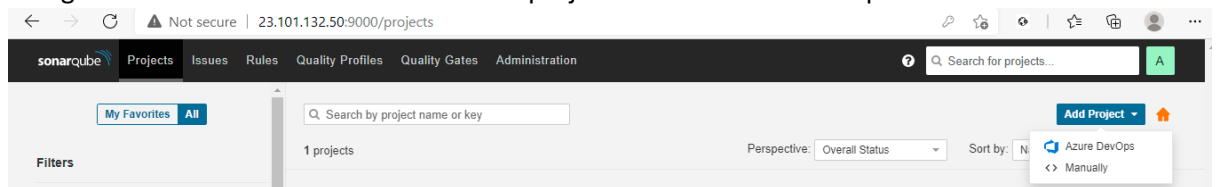
8. Now, Navigate to Azure DevOps portal and create PAT.
9. Navigate to sonarqube and Administration>ALM integration>Azure DevOps>give all the details as asked.



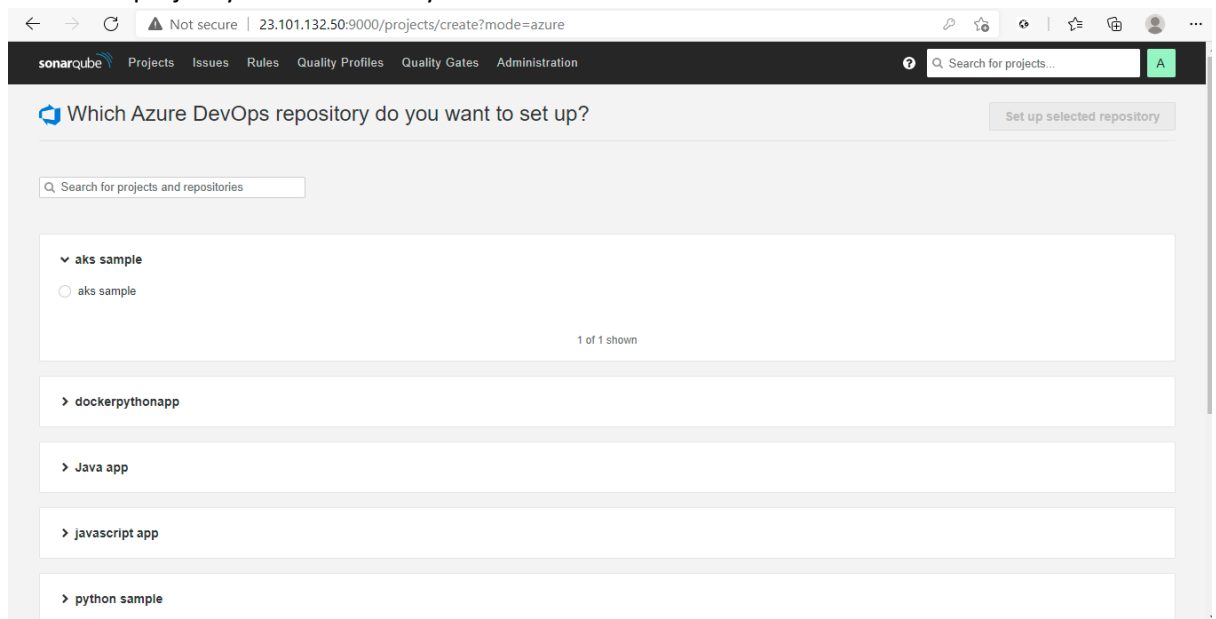
10. And once done, you will be able to check the configuration.



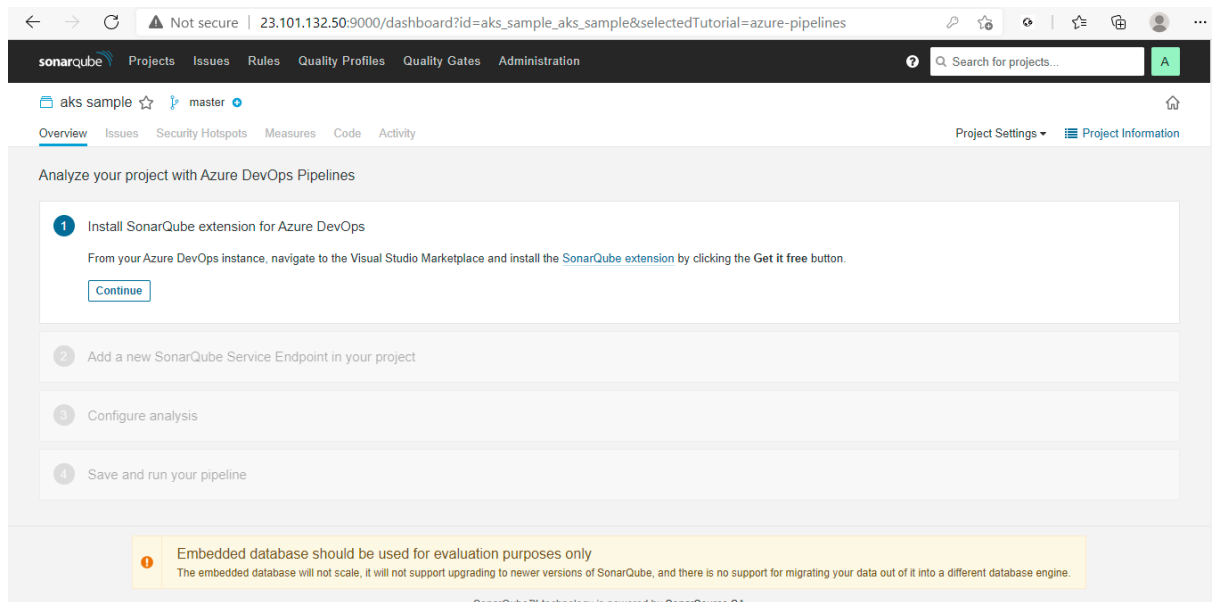
11. Now you are set to integrate your organization projects to sonarqube for code analysis.
12. Navigate to the dashboard and under “Add project”>select “Azure Devops”.



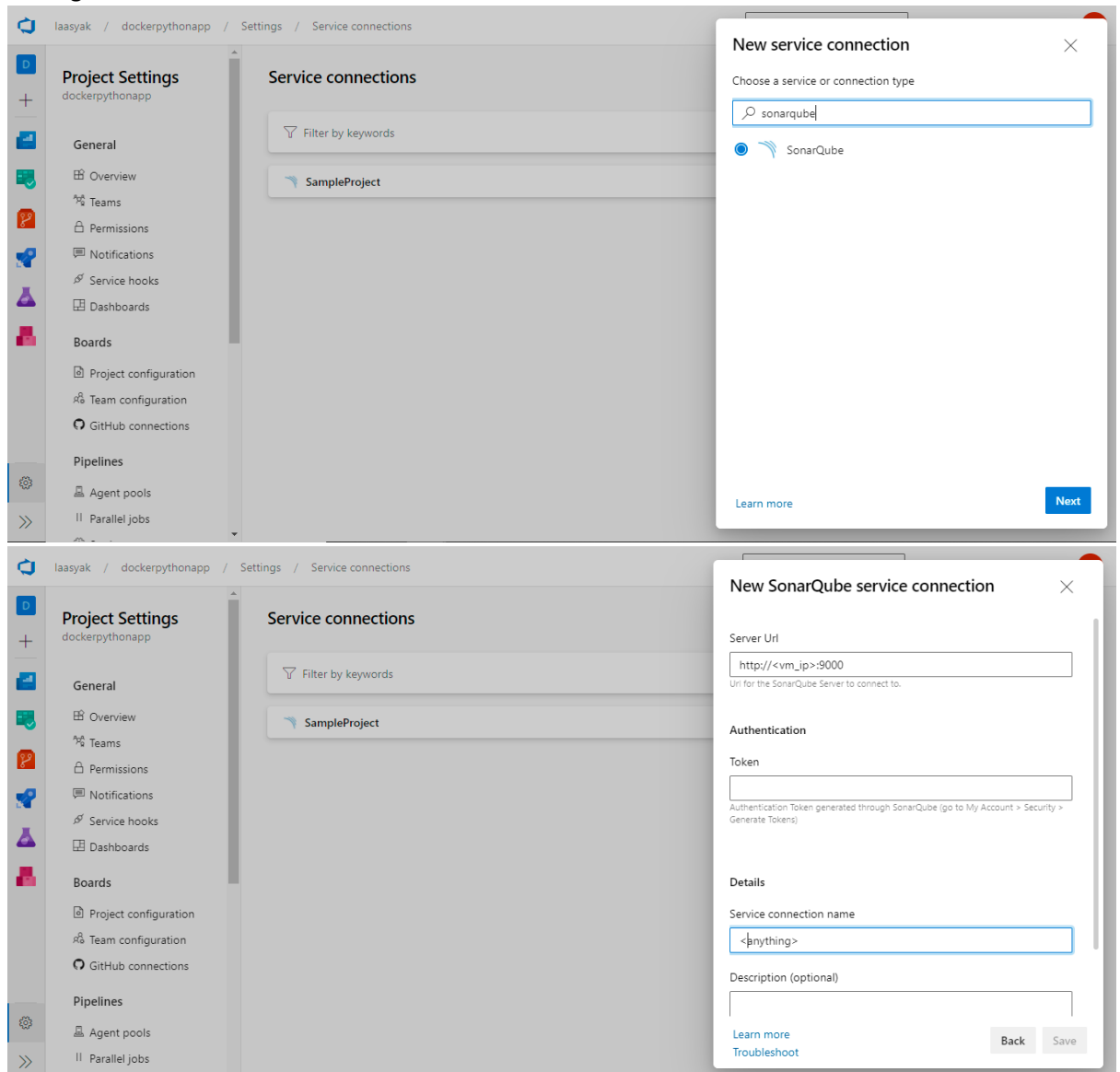
13. Select the project you want to analyse.



14. Select that and Set up the selected repository. Once done, it will provide the steps to follow.



15. In the azure devops portal, you will have to create a service connection under your project settings.



16. To add the token value in the connection you want to create, navigate to the sonarqube dashboard. Myaccount>security>generate token. Copy paste it to the field and save it.
17. Once done with all of this, go to the repo section of devops project and import a repo of any project you wish to build.
18. Build the pipeline by settings as specified in the following screenshots.

The first screenshot shows the 'Prepare Analysis Configuration' task configuration. The task version is 4.*. The display name is 'Prepare analysis on SonarQube'. The SonarQube Server Endpoint is 'SampleProject'. The way to run the analysis is 'Integrate with Maven or Gradle'. The advanced options are expanded, showing 'Control Options' with 'Enabled' checked and 'Continue on error' unchecked. The 'Additional Properties' section contains the following text:

```
# Additional properties that will be passed to the scanner.
# Put one key=value per line, example:
# sonar.exclusions=**/*.bin
sonar.projectKey=dockerpythonapp_dockerpythonapp
```

The second screenshot shows the 'Maven pom.xml' task configuration. The task version is 4.*. The display name is 'Maven pom.xml'. The SonarQube Server Endpoint is 'SampleProject'. The way to run the analysis is 'Integrate with Maven or Gradle'. The advanced options are expanded, showing 'Control Options' with 'Enabled' checked and 'Continue on error' unchecked. The 'Additional Properties' section contains the following text:

```
# Additional properties that will be passed to the scanner.
# Put one key=value per line, example:
# sonar.exclusions=**/*.bin
sonar.projectKey=dockerpythonapp_dockerpythonapp
```

The third screenshot shows the 'Maven pom.xml' task configuration. The task version is 4.*. The display name is 'Maven pom.xml'. The SonarQube Server Endpoint is 'SampleProject'. The way to run the analysis is 'Integrate with Maven or Gradle'. The advanced options are expanded, showing 'Control Options' with 'Enabled' checked and 'Continue on error' unchecked. The 'Additional Properties' section contains the following text:

```
# Additional properties that will be passed to the scanner.
# Put one key=value per line, example:
# sonar.exclusions=**/*.bin
sonar.projectKey=dockerpythonapp_dockerpythonapp
```

19. Now run the pipeline.

20. Once succeeded, navigate to sonarqube and you will be able to see the insights of your pipeline job.

The screenshot displays the SonarQube web interface. The top navigation bar includes 'sonarqube', 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. A search bar is present on the right. The main content area shows a list of projects, with 'petclinic' selected and marked as 'Passed'. The project details panel on the right provides a comprehensive overview of the analysis results:

- Reliability (Bugs):** 174 bugs (Grade E)
- Vulnerabilities:** 9 vulnerabilities (Grade D)
- Hotspots Reviewed:** 0.0% (Grade E)
- Code Smells:** 1.1k (Grade A)
- Coverage:** 0.0% (Grade F)
- Duplications:** 65.4% (Grade F)
- Lines:** 55k (Grade M)

A warning message states: 'Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.'

The bottom section of the image shows the 'petclinic' project page with a green 'Passed' status. The 'Overview' tab is active, displaying a summary of the project's quality metrics:

- New Code:** 174 Bugs (Grade E)
- Overall Code:** 9 Vulnerabilities (Grade D)
- Security Hotspots:** 6 (Grade E), 0.0% Reviewed
- Debt:** 73d (Grade D), 1.1k Code Smells (Grade A)

The bottom status bar indicates the user is logged in as 'admin'.