

# API Automation with Postman-Newman & k6, CI/CD in Azure

---

## Step 1 Create a VSCode Project

1. Open **Visual Studio Code**.
2. Create a new folder for your project, e.g.:

```
bash
CopyEdit
mkdir arctic-api-automation
cd arctic-api-automation
```

3. Open the folder in VSCode:

```
css
CopyEdit
code .
```

4. Initialize Node.js project:

```
bash
CopyEdit
npm init -y
```

This creates a `package.json`.

---

## Step 2 Recommended Project Folder Hierarchy

```
pgsql
CopyEdit
arctic-api-automation/
├── postman-collections/
│   └── ArcticAPI.postman_collection.json
├── k6-scripts/
│   └── arctic_api_loadtest.js
├── newman-reports/                                # (Generated reports)
├── cicd-pipelines/
│   └── azure-pipelines.yml
├── package.json
├── README.md
└── .gitignore
```

- `postman-collections/` → Your Postman collection `.json` files.
- `k6-scripts/` → Your k6 load test scripts.
- `newman-reports/` → Output reports generated by Newman & k6.

- `cicd-pipelines/` → CI/CD pipeline files (like YAML).
- 

## Step 3 Install Dependencies

✓ Install Newman globally:

```
bash
CopyEdit
npm install -g newman
```

✓ (Optional) Install locally & add as dev dependency:

```
bash
CopyEdit
npm install --save-dev newman
```

✓ Install k6 on your machine:

- Mac:

```
bash
CopyEdit
brew install k6
```

- Windows/Linux:  
See k6 installation docs
- 



## Step 4 Write Postman Collection

Export your Postman collection from the Postman app, save as:

```
pgsql
CopyEdit
postman-collections/ArcticAPI.postman_collection.json
```

✓ Run it with Newman:

```
bash
CopyEdit
newman run postman-collections/ArcticAPI.postman_collection.json -r
cli,html --reporter-html-export=newman-reports/report.html
```

---



## Step 5 Write k6 Script

```
k6-scripts/arctic_api_loadtest.js
```

```
javascript
CopyEdit
import http from 'k6/http';
import { check, sleep } from 'k6';

export const options = {
  vus: 10,
  duration: '30s',
};

export default function () {
  const res = http.get('https://your-api-endpoint.com/api');
  check(res, { 'status is 200': (r) => r.status === 200 });
  sleep(1);
}
```

✅ Run it:

```
bash
CopyEdit
k6 run k6-scripts/arctic_api_loadtest.js --summary-export=newman-
reports/k6-summary.json
```

---

## Step 6 Add Scripts in `package.json`

```
json
CopyEdit
"scripts": {
  "test:api": "newman run postman-
collections/ArcticAPI.postman_collection.json -r cli,html --reporter-html-
export=newman-reports/report.html",
  "loadtest": "k6 run k6-scripts/arctic_api_loadtest.js --summary-
export=newman-reports/k6-summary.json"
}
```

✅ Run via:

```
bash
CopyEdit
npm run test:api
npm run loadtest
```

---

## Step 7 Create Azure DevOps Pipeline

✅ Add pipeline YAML in:

```
bash
CopyEdit
ci/cd-pipelines/azure-pipelines.yml
```

Sample:

```
yaml
```

```

CopyEdit
trigger:
- main

pool:
  vmImage: 'ubuntu-latest'

steps:
- task: NodeTool@0
  inputs:
    versionSpec: '18.x'
    displayName: 'Install Node.js'

- script: |
  npm install -g newman
  npm install
  displayName: 'Install Newman'

- script: |
  newman run postman-collections/ArcticAPI.postman_collection.json -r
  cli,html --reporter-html-export=newman-reports/report.html
  displayName: 'Run Postman Collection with Newman'

- script: |
  sudo apt-get install -y gnupg software-properties-common
  sudo apt-get update
  sudo apt-get install -y k6
  displayName: 'Install k6'

- script: |
  k6 run k6-scripts/arctic_api_loadtest.js --summary-export=newman-
  reports/k6-summary.json
  displayName: 'Run k6 Load Test'

- task: PublishBuildArtifacts@1
  inputs:
    PathToPublish: 'newman-reports'
    ArtifactName: 'API_Test_Reports'
    publishLocation: 'Container'

```

- ✓ Commit & push to Azure DevOps repository.
- ✓ Create a pipeline, point to this YAML, and run.

## Step 8 Output

### ✓ Pipeline Artifacts:

- Functional test report (Newman) → report.html
- Load test summary (k6) → k6-summary.json

- ✓ Reports are saved under `newman-reports/` and published as pipeline artifacts.

## ★ Summary:

- ✓ Create VSCode project →
- ✓ Organize folders →
- ✓ Install Newman & k6 →
- ✓ Write Postman collection →
- ✓ Write k6 script →
- ✓ Run locally →
- ✓ Integrate with Azure Pipeline.

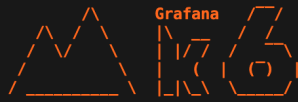
Execute the command

Execution output.

Post Man Execution Result

	executed	failed
iterations	1	0
requests	17	0
test-scripts	10	0
prerequest-scripts	0	0
assertions	20	3
total run duration: 6.8s		
total data received: 18.8KB (approx)		
average response time: 380ms [min: 228ms, max: 1177ms, s.d.: 246ms]		

## K6 Load Testing Summary report



execution: local  
script: ./k6-scripts/api\_load\_test.js  
output: -

scenarios: (100.00%) 1 scenario, 10 max VUs, 1m0s max duration (incl. graceful stop):  
\* default: 10 looping VUs for 30s (gracefulStop: 30s)

✓ status was 200  
✓ response time < 500ms

```
checks.....: 100.00% 480 out of 480
data_received.....: 1.3 MB 44 kB/s
data_sent.....: 34 kB 1.1 kB/s
http_req_blocked.....: avg=29.71ms min=1µs med=9µs max=724.31ms p(90)=17.1µs p(95)=34.64µs
http_req_connecting.....: avg=9.24ms min=0s med=0s max=227.04ms p(90)=0s p(95)=0s
http_req_duration.....: avg=226.11ms min=221.62ms med=225.59ms max=233.79ms p(90)=229.97ms p(95)=230.78ms
{ expected_response:true }...: avg=226.11ms min=221.62ms med=225.59ms max=233.79ms p(90)=229.97ms p(95)=230.78ms
http_req_failed.....: 0.00% 0 out of 240
http_req_receiving.....: avg=157.1µs min=20µs med=107.5µs max=1.36ms p(90)=224.1µs p(95)=316.54µs
http_req_sending.....: avg=77.32µs min=3µs med=24µs max=2.03ms p(90)=62.1µs p(95)=133.05µs
http_req_tls_handshaking.....: avg=19.48ms min=0s med=0s max=474.71ms p(90)=0s p(95)=0s
http_req_waiting.....: avg=225.87ms min=221.49ms med=225.39ms max=233.67ms p(90)=229.84ms p(95)=230.55ms
http_reqs.....: 240 7.921715/s
iteration_duration.....: avg=1.25s min=1.22s med=1.22s max=1.95s p(90)=1.23s p(95)=1.23s
iterations.....: 240 7.921715/s
vus.....: 10 min=10 max=10
vus_max.....: 10 min=10 max=10
```