Solana Anchor DevKit: VS Code in the Cloud

This AMI is fundamentally a **Solana test-validator sandbox** in the cloud.

Usage instructions:

1. Launch the product via 1-click from AWS Marketplace. **Wait** until the instance status changes to 'Running' and passes all health checks. Then, connect to your instance using your Amazon private key and the '**ubuntu**' user."

To update software, use: **sudo apt update && sudo apt upgrade -y**

Build the Docker image

From your home directory on the fresh EC2 instance:

```
cd ~/solana-sandbox
```

Build the “solana-sandbox” image

```
docker build -t solana-sandbox .
```
(include period)

Launch the sandbox container

Run in “host” networking mode so RPC (8899) and VS Code (8080) are exposed directly

```
docker rm -f solana-sandbox 2>/dev/null || true

docker run -d \
  --name solana-sandbox \
  --network host \
  -e PASSWORD="YourStrongPassword" \
  -v ~/solana-sandbox/workspace:/workspace \
  solana-sandbox
```

- **PASSWORD** sets your VS Code login password
- **/workspace** is where you’ll create and edit your Anchor projects

Open VS Code Server

In your browser, go to

- http://Your_instance_public_IP**:8080**
- Log in with “YourStrongPassword” from above

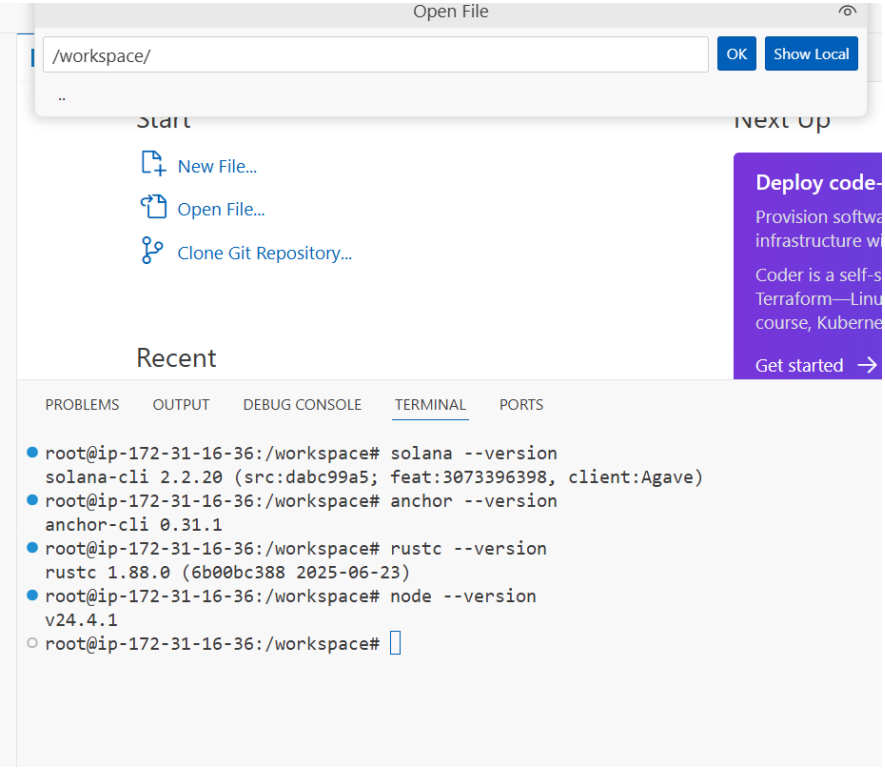
In the Explorer pane, choose **Open Folder** → navigate to **/workspace**

In the input box type : /workspace

The on terminal > New Terminal

<u>Smoke-test the Solana + Anchor toolchain</u>
Inside the VS Code Server integrated terminal:

<u>Verify versions</u>
solana --version
anchor --version
rustc --version



Point Anchor at the local validator, run in the workspace terminal:

solana config set --url http://localhost:8899

That tells Anchor (and the Solana CLI) to talk to your in-container solana-test-validator instead of the public mainnet. Once you see the new RPC URL printed, you're ready to anchor build, anchor deploy, and anchor test against your local sandbox.

Quickly scaffold and test a program
In same directory: cd /workspace

anchor init quickstart --javascript
cd quickstart

anchor build # compile your "Hello World"
anchor deploy # deploy to the localnet
anchor test # run the built-in test suite

If all of these succeed, your Dockerized Solana + Anchor + VS Code sandbox is up and running!

****(Optional) Bring up your front-end with Docker Compose**
If you want to add a frontend, write your script here:

cd ~/solana-sandbox
nano docker-compose.yml

PASSWORD="YourStrongPassword" docker-compose up -d

This will:

- Start your solana-sandbox container
- Build & start your frontend container (on port 3000, if configured)

For Help: https://docs.solana.com/cli/install-solana-cli-tools

Help Documentation:

This AMI is fundamentally a **Solana test-validator sandbox** in the cloud. Here's how it lets you write, deploy, and run Solana programs end-to-end:

---

1. A Local Solana Cluster, Always Running

- When you start the Docker container (or with docker-compose up), our start.sh does:

solana-test-validator --reset --limit-ledger-size --faucet-lamports 1000000000000 &

- That spins up a full **local** Solana node on localhost:8899 with its own ledger and a built-in "faucet" so you can airdrop yourself tokens whenever you need them.

In Practice, You "Test Solana" by…

1. **Writing** a smart contract in programs/ under /workspace.
2. **Building** & **deploying** it with Anchor, which targets your sandbox node.
3. **Executing** transactions or running integration tests (anchor test) all on that isolated, restartable local network.
4. **Debugging** interactively in VS Code (breakpoints, logs, on-chain state via CLI).

No external network calls, no local installs—just a **self-contained**, cloud-hosted Solana development & test environment that mirrors mainnet behavior without the risk or latency.

AWS Data

- Data Encryption Configuration: This solution does not encrypt data within the running instance.
- User Credentials are stored: /root/.ssh/authorized_keys & /home/ubuntu/.ssh/authorized_keys
- Monitor the health:
    - Navigate to your Amazon EC2 console and verify that you're in the correct region.
    - Choose Instance and select your launched instance.
    - Select the server to display your metadata page and choose the Status checks tab at the bottom of the page to review if your status checks passed or failed.

Extra Information: **(Optional)**

Allocate Elastic IP

To ensure that your instance **keeps its IP during restarts** that might happen, configure an Elastic IP. From the EC2 console:

1. Select ELASTIC IPs.
2. Click on the ALLOCATE ELASTIC IP ADDRESS.
3. Select the default (Amazon pool of IPv4 addresses) and click on ALLOCATE.
4. From the ACTIONS pull down, select ASSOCIATE ELASTIC IP ADDRESS.
5. In the box that comes up, note down the Elastic IP Address, which will be needed when you configure your DNS.
6. In the search box under INSTANCE, click and find your INSTANCE ID and then click ASSOCIATE.
7. Your instance now has an elastic IP associated with it.
8. For additional help: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html