Contents

1 Solana Containerised Testbed Program **1**
1.1 Current validated state . . . 1
1.2 Publications and identifiers . . . 2
1.3 Repository structure . . . 2
1.4 Container images . . . 3
1.5 Why no-AVX2? . . . 3
1.6 Local Compose-based testbed . . . 3
1.7 Yellowstone/Geyser release mode . . . 3
1.8 Kubernetes Observability Core . . . 4
1.9 Dedicated KVM multi-node Kubernetes deployment . . . 4
1.10 Wallet transfer validation . . . 5
1.11 Research roadmap . . . 5
1.12 Citation . . . 6
1.13 License . . . 6

1 Solana Containerised Testbed Program

Containerised and Kubernetes-oriented Solana local testbed for reproducible infrastructure, observability, dataset generation, and latency/performance research on legacy x86_64 hardware.

The project started as a local containerised Solana v1.18.25 testbed and has evolved into a Kubernetes-based observability environment. The current validated stage extends the previous Minikube validation to a dedicated KVM-based multi-node Kubernetes cluster.

1.1 Current validated state

The current development state is `v0.4.0` preparation.

The latest validated deployment is:

`Dedicated KVM Multi-Node Kubernetes Deployment of the Solana Kubernetes Observability Core`

This stage validates the existing Kubernetes Observability Core on a dedicated kubeadm cluster running on KVM virtual machines.

Validated capabilities include:

- Solana v1.18.25 local validator;
- no-AVX2 compatibility image for older x86_64 CPUs;
- Yellowstone/Geyser-enabled validator image;
- containerised latency monitor;
- Kubernetes validator StatefulSet;
- validator ledger PersistentVolumeClaim;
- wallet-init Job;
- monitor Deployment;
- RPC Service on port `8899`;
- Yellowstone/Geyser gRPC Service on port `10000`;
- metrics Service on port `9464`;
- wallet transfer validation through the Kubernetes RPC endpoint;

- transaction latency telemetry exposed through the metrics endpoint.

The `v0.4.0` stage remains focused on infrastructure and observability. It does not introduce controlled load generation, dashboarding, MPC, reinforcement learning, MARL, or Agave.

1.2 Publications and identifiers

Software identifiers:

- Software concept DOI: `10.5281/zenodo.20095383`
- Software v0.1.x DOI: `10.5281/zenodo.20095384`
- Software v0.2.0 DOI: `10.5281/zenodo.20132465`
- Software v0.3.0 DOI: `10.5281/zenodo.20337744`
- Software v0.4.0 DOI: `10.5281/zenodo.20551170`

Technical reports and notes:

- Containerising a Solana v1.18.25 Local Testbed for Reproducible Dataset Generation on Legacy x86_64 CPUs: https://doi.org/10.5281/zenodo.20098291
- Yellowstone/Geyser Observability Extension for the Solana Containerised Testbed v0.2.0: https://doi.org/10.5281/zenodo.20167936
- From Compose to Kubernetes: Validating the Observability Core of a No-AVX2 Solana Containerised Testbed: https://doi.org/10.5281/zenodo.20340310
- Dedicated KVM Multi-Node Kubernetes Deployment technical report: https://doi.org/10.5281/ze

1.3 Repository structure

```
.
├── compose.yaml
├── compose.release.yaml
├── compose.yellowstone.release.yaml
├── docs
│   ├── kubernetes
│   ├── kvm-multinode-observability-core
│   ├── releases
│   └── research-program
├── infra
│   └── ansible
│       └── kvm-kubeadm
├── k8s
│   ├── base
│   └── overlays
│       ├── minikube
│       └── kvm-multinode
├── monitor
├── results
│   ├── kvm-kubeadm-cluster
│   └── kvm-multinode-observability-core
├── scripts
├── validator
└── wallet-init
```

1.4 Container images

The current published images used by the Kubernetes Observability Core are:

- `docker.io/khoshaba/solana-localnet-validator:v1.18.25-noavx2-ivybridge-yellowstone`
- `docker.io/khoshaba/solana-wallet-init:v1.18.25-noavx2-ivybridge`
- `docker.io/khoshaba/solana-latency-monitor:v0.2.0`

1.5 Why no-AVX2?

Some older x86_64 CPUs do not support AVX2. On such machines, standard prebuilt Solana validator binaries may fail with:

```
Illegal instruction
Aborted
exit code 139
```

For this reason, the project uses source-built no-AVX2 compatible Solana images for the target legacy hardware.

See:

- `docs/compatibility.md`
- `docs/troubleshooting.md`

1.6 Local Compose-based testbed

Copy the environment file:

```
cp .env.example .env
```

Start the validator and wallet bootstrap:

```
podman compose up validator wallet-init
```

Check RPC health:

```
curl -s http://127.0.0.1:8899 \
  -H "Content-Type: application/json" \
  -d '{"jsonrpc":"2.0","id":1,"method":"getHealth"}'
```

Expected result:

```
{"jsonrpc":"2.0","result":"ok","id":1}
```

Stop the local testbed:

```
podman compose down
```

Remove the local ledger volume:

```
podman compose down -v
```

1.7 Yellowstone/Geyser release mode

Run the Yellowstone/Geyser-enabled release workflow:

```
podman compose -f compose.yellowstone.release.yaml up
```

Check validator health:

```
curl -s http://127.0.0.1:8899 \
  -H "Content-Type: application/json" \
  -d '{"jsonrpc":"2.0","id":1,"method":"getHealth"}'
```

Check Prometheus metrics:

```
curl -s http://127.0.0.1:9464/metrics | head -n 40
```

Immutable image digests for the v0.2.0 Yellowstone/Geyser stage are listed in:

- `release/v0.2.0-image-digests.txt`

1.8 Kubernetes Observability Core

The portable Kubernetes Observability Core is stored under:

- `k8s/base/`

It includes:

- `solana-observability` namespace;
- validator StatefulSet;
- validator ledger PVC;
- wallet-init Job;
- monitor Deployment;
- RPC Service on port `8899`;
- Yellowstone/Geyser gRPC Service on port `10000`;
- metrics Service on port `9464`.

The Minikube validation overlay is stored under:

- `k8s/overlays/minikube/`

The dedicated KVM multi-node overlay is stored under:

- `k8s/overlays/kvm-multinode/`

The KVM multi-node overlay adds:

- validator placement on `testbed-role=validator`;
- monitor placement on `testbed-role=observability`;
- explicit `local-path` storage for the validator ledger PVC.

1.9 Dedicated KVM multi-node Kubernetes deployment

The dedicated deployment uses a three-node kubeadm cluster:

Node	Role	Label
`k8s-cp-01`	Kubernetes control-plane	`testbed-role=control-plane`
`k8s-worker-01`	Solana validator workload	`testbed-role=validator`
`k8s-worker-02`	Observability workload	`testbed-role=observability`

The validated infrastructure stack includes:

- CentOS Stream 9 KVM/libvirt host;
- Ubuntu Server 24.04 LTS guest VMs;
- containerd runtime;
- kubeadm bootstrap;
- Calico CNI;
- local-path-provisioner;
- Ansible automation for preparation, bootstrap, validation, and reset.

The Ansible automation is stored under:

- `infra/ansible/kvm-kubeadm/`

The validation evidence is stored under:

- `results/kvm-kubeadm-cluster/`
- `results/kvm-multinode-observability-core/`

1.10 Wallet transfer validation

The wallet transfer validation script is:

- `scripts/k8s-wallet-transfer-validation.sh`

It creates a Kubernetes Job that:

- checks the Solana CLI version;
- checks the Solana cluster version;
- creates a temporary sender keypair;
- airdrops `2 SOL` to the sender;
- transfers `0.5 SOL` to the receiver;
- confirms the transaction;
- prints final sender and receiver balances.

In the validated KVM multi-node deployment, the transfer completed successfully and the metrics endpoint reported a transaction latency sample.

1.11 Research roadmap

The project follows a layered research methodology:

```
observation -> controlled load -> model -> MPC -> single-agent RL -> MARL
```

The current `v0.4.0` stage belongs to:

```
infrastructure -> observability
```

The following remain out of scope for the current stage:

- synthetic load generation;
- dashboard development;
- Prometheus/Grafana stack;
- MPC;
- single-agent RL;
- MARL;
- Agave.

Agave remains a separate research branch.

See:

- `docs/research-program/`

1.12 Citation

Citation metadata is provided in:

- `CITATION.cff`
- `.zenodo.json`

The software and technical report identifiers will be updated after the `v0.4.0` Zenodo publication.

1.13 License

See `LICENSE`.