

Project Talan: CLI

pitch deck

A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.

Talan CLI

Talan CLI (tln) is an open-source framework for managing third-party components from wide range of ecosystems (Java, Node.js, C++, Golang, Angular etc.).

tln helps to create fully isolated development environments,

uniformly manages mono- & multi- repo configurations,

builds smooth onboarding experience,

melts borders between local development environments and CI/CT/CD setups,

gets maximum from Polyglot Programming Polyglot Persistence (4Ps) design,

implements Software Architecture as Code

Projects cataloging

- Single point of truth for all company projects, internal or/and external
- Painless switch between projects
- Handle mono-/multi- repo configurations

```
/
└─ petramco
   │ delivery
   │ │ hyperhype
   │ │ │ cms
   │ │ │ crm
   │ │ │ portal
   │ │ └─ warehouse
   │ └─ skeletons
   │   │ java
   │   │ nodejs
   │   │ rust
   │   └─ typescript
   │ hr
   │ └─ referral-grid
   └─ it
       └─ web
           │ brand-site
           │ portal
           └─ solutions
```

Uniform Dev/CI implementation

- All key development steps should have the same structure everywhere.
- CI acts as a “silly developer” and implements just CI specific steps: scheduling, vault, execute PR/PUSH/NIGHTLY builds

```
root@devbox:~# tlh run testing
> Using Docker
- Your Docker API version is 1.40
> Downloading Selenoid...
- Fetching tags for image aerokube/selenoid
registry.ping url=https://registry.hub.docker.com/v2/
registry.tags url=https://registry.hub.docker.com/v2/aerokube/se
- Pulling image aerokube/selenoid:1.10.3
> Configuring Selenoid...
> Requested to sync configuration from "./browsers.json"...
- Pulling image selenoid/vnc:chrome_90.0
```

```
312 |  
313 |         tln install testing --depends  
314 |         tln run testing  
315 |     displayName: Run  
316 |
```

Cross –platform/–ecosystem virtual environment

- Manage different versions of components (java, kubectl, helm, angular, go, etc.)
- Handle multiple environments at the same dev box
- Quick project/developer onboarding
- Don't modify underlying OS

```
module.exports = {
  options: async (tln, args) => {},
  env: async (tln, env) => {},
  dotenvs: async (tln) => [],
  inherits: async (tln) => [],
  depends: async (tln) => ['mvn-3.6.3', 'openjdk-11.0.2', 'go-1.14.4', 'node-14.4.0', 'angular-9'],
  steps: async (tln) => [
    {
      id: "versions",
      builder: async (tln, script) => script.set([
        'java -version && mvn -v && go version && node -v && cordova -v && ng version'
      ])
    }
  ],
  components: async (tln) => []
}
```

```
openjdk version "11.0.2" 2019-01-15
OpenJDK Runtime Environment 18.9 (build 11.0.2+9)
OpenJDK 64-Bit Server VM 18.9 (build 11.0.2+9, mixed mode)
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: /root/work/maven/mvn-3.6.3
Java version: 11.0.2, vendor: Oracle Corporation, runtime: /root/.sdkman/candidates/java/11.0.2-openjdk
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "4.15.0-143-generic", arch: "amd64",
go version go1.14.4 linux/amd64
v14.4.0
9.0.0 (cordova-lib@9.0.1)
```

Software Architecture as Code (tln-cli-2.x.x)

- Maintain consistent relation between source code and UML/C4 diagrams
- Every PR will show changes not only at source code level, but also unhide changes in architecture

