



The Secret to Managing Shared Secrets



ActiveState “State Tool” Webinar

Welcome



Pete Garcin

Senior Product Manager
ActiveState (@rawktron)



Dana Crane

Product Marketing
Manager, ActiveState

Housekeeping

- Webinar recording and slides will be available shortly
- Share questions with panelists using the Question panel
- Q&A session following presentations

About ActiveState

- **Track-record:** 97% of Fortune 1000, 20+ years open source
- **Polyglot:** 5 languages - Python, Perl, Tcl, Go, Ruby
- **Runtime Focus:** concept to development to production



vmware®

NORTHROP GRUMMAN

***Rockwell
Collins***



Alcatel • Lucent



SIEMENS

ActiveState®

About ActiveState Platform

- **Runtimes:** automatically builds runtime environments in minutes
- **Dependency Management:** automatically pulls in & resolves all dependencies
- **Multilingual; Multiplatform:** automatically packages Python & Perl runtimes for Windows & Linux

Overview

- Activating a Project
- Project Configuration
- Constants & Secrets
- Scripts & Events
- Real World: Project Setup
- Q&A




ActiveState Platform


What if we could reduce your entire development environment setup to a single command?

ActiveState Platform

- Sign up now for an account and to download the state tool:
 - <https://platform.activestate.com>

The Secret to Managing Shared Secrets

ActiveState Your Dashboard Featured Projects & Languages **Dev Tools**   Kinman 

Dev Tools 

The State Tool

A CLI tool that puts the power of the ActiveState Platform right in your terminal.

The State Tool is in a preview period and works on Windows and Linux only. We're actively changing and adding features based on developer feedback.



[Install on Windows](#) [Install on Linux](#)

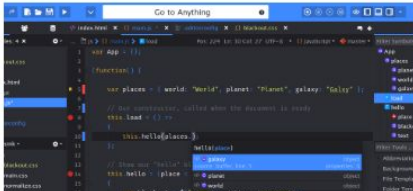
```
powershell "IEX(New-Object Net.WebClient).GetContent('https://aka.ms/activestate-devtools')
```

Needs to be run with administrator privileges.

Komodo IDE

Our multi-language IDE is now available as part of your ActiveState Platform subscription and, for a limited time, it is free to all users.

[Download](#)  [Learn more](#) 



ActiveState®

Installing the State Tool

- Currently the public release of the state tool only supports Linux & Windows. macOS support will follow in the not so distant future.
- To install the State tool simply run the following one liner from your command prompt:

```
sh <(curl -q https://platform.activestate.com/dl/cli/install.sh)
```

```
powershell "IEX(New-Object Net.WebClient).downloadString('https://platform.activestate.com/dl/cli/install.ps1')"
```

The Secret to Managing Shared Secrets

The screenshot shows the ActiveState web interface. The top navigation bar includes 'ActiveState', 'Your Dashboard', 'Featured Projects & Languages', 'Dev Tools' (which is highlighted with a green arrow), and a user profile 'Kinman'. Below the navigation bar is a 'Dev Tools' header. The main content area is divided into two columns. The left column is titled 'The State Tool' and describes it as a CLI tool. It includes a yellow callout box stating it's in a preview period and works on Windows and Linux only. Below this are buttons for 'Install on Windows' and 'Install on Linux'. A code snippet for PowerShell is shown: `powershell "IEX(New-Object Net.W...". A yellow circle highlights a copy icon next to the code. The right column is titled 'Komodo IDE' and describes it as a multi-language IDE. It includes 'Download' and 'Learn more' buttons. Below the text is a screenshot of the Komodo IDE interface.`

ActiveState Your Dashboard Featured Projects & Languages **Dev Tools** Kinman

Dev Tools

The State Tool

A CLI tool that puts the power of the ActiveState Platform right in your terminal.

The State Tool is in a preview period and works on Windows and Linux only. We're actively changing and adding features based on developer feedback.

[Install on Windows](#) [Install on Linux](#)

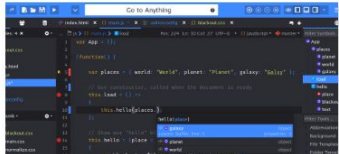
```
powershell "IEX(New-Object Net.W...
```

Needs to be run with administrator privileges.

Komodo IDE

Our multi-language IDE is now available as part of your ActiveState Platform subscription and, for a limited time, it is free to all users.

[Download](#) [Learn more](#)



- 1) Go back to the platform
- 2) Click the “copy” button
- 3) Go back to the Command Prompt
- 4) CTRL-V/Right click to paste

ActiveState

Authenticating

- The state tool prompts you to enter your username and password automatically if you're not already authenticated.
- However, if you ever need to manually authenticate, run the following command to authenticate your CLI:

```
state auth
```

- You will be prompted for your username and password, and if all goes well it should show a friendly “You have authenticated” message.

Activating your Project

- Now, we simply activate our project with the following command:

```
state activate owner/projectName
```

- Owner can be either username or organization name. For instance, if you created your project inside an organization, then use the organization name instead of your username.

ActiveState Platform

- That's it! One line has:
 - Created a virtual environment for your project
 - Installed the Python runtime and all your defined project dependencies
 - As we'll see, it can also run subsequent commands, configuration and setup.
 - The ActiveState platform will keep all your dependencies updated and current if you've set them up that way.

Managing Secrets and Project Config

- The state tool doesn't just do runtime environment management, but also lets you set up your development environment, automate workflows and share secrets with your team.

Project Configuration

- Inside your repo, you configure your project using the `activestate.yaml` file.
- Inside this file you can define:
 - Your project URL that tells the platform what runtime to use
 - Constants -- values that you wish to use in scripts
 - Scripts -- scripts written in a language of your choice that help automate your workflow

Static Values (Constants)

- Let's start with the simplest: Defining static values. Open up the activestate.yaml file that was created under your project directory. Let's define a simple constant:

```
constants:  
  - name: LOCATION  
    value: World
```

Static Values

- You now have a constant defined that you can use throughout your config.
- Let's try actually using it though, add another constant:

```
- name: HELLO  
  value: Hello $constants.LOCATION
```

Secrets

- You probably have a bunch of shared credentials, API Keys, etc. how are you storing these?
 - Wiki?
 - External tool like vault, etc.?
 - Slack/Email?
 - 1Password/LastPass?
 - GitHub!? *gasp*

Managing Secrets & Project Config

- Secrets have the concept of scopes -- which allow you to automatically share them amongst all members of the scope.
- For example, if you set a secret to have project level scope, everyone within that project will have access to that secret.
- Compare that to a user level secret, where only you have access to the value of that secret.
 - Other users will still have access to the secret name -- but will set their own value for that user-level secret

Defining Secrets

- For now, we define secrets using the tool and not in the activestate.yaml.
- In the future you will be able to define secrets in the activestate.yaml file as well.

```
state secrets set project.secretname value
```

Defining Secrets

- To create a secret value that only you have access to:

```
state secrets set user.LOCATION value
```

- Reminder: This will still define the secret for everyone on the project, but only you will have access to the value you've set. Anyone else that uses this secret will be prompted for their own value.

Defining Secrets



Scripts
Constants
Secrets
Events

Secrets

Secrets are client-side encrypted values that can be shared between your environments and people working on this project. Use them in your [Scripts](#) for things like API keys or database passwords.

> [Setting up to use secrets](#)

[Add a Secret](#)

Secrets in this project

Name	Type	Description
> world	User	A description was not provided.
> test	User	A description was not provided.

To view secrets on the command line, run `> state secrets`

Retrieving Secrets

```
state secrets get user.LOCATION
```

- ..or if you want to use it in your activestate.yaml

```
constants:  
  - name: HELLO  
    value: Hello $secrets.user.LOCATION
```

Automating Workflows: Scripts

- The real power of the state tool starts to become apparent when you leverage it to automate configuration and workflows.

```
scripts:  
- name: simpleHello  
  value: echo This is a simple Hello World script.
```

Scripts: Using Constants & Secrets

- Scripts can also use constants, so we can embed one of our earlier constants:

```
value: echo $constants.HELLO
```

- This will work for any type of field, including secrets.

Scripts: Nesting Scripts

- It gets more interesting though, because in the activestate.yaml EVERYTHING can be used as a variable, so you could create another script that references our first script:

```
scripts:  
- name: log-hello  
  value: $scripts.hello > /tmp/hello.txt
```

Scripts: Arguments

- You can also forward any arguments from command line invocation to your scripts to make them even more flexible. So in this case, if we execute ``state run arg-hello World`` with the below script defined, our output will be: “Hello World”

```
scripts:  
- name: arg-hello  
  value: echo Hello $1
```

Automating Config: Events

- Events act mostly the same as scripts do, except that they aren't manually invoked and instead run when their event triggers. For example we could have an ACTIVATE event that looks like this:

```
events:  
- name: ACTIVATE  
  value: systemctl start my-service
```

- This would start a service whenever we enter an “activated state”. It's worth noting that the ACTIVATE event has a special use-case: it is invoked as part of your bashrc (or zshrc, or fishrc, or ..) meaning it can export environment variables, register bash aliases, etc.

Real World: Project Setup

- Let's take a look at setting up a real world -- existing project for use with the state tool.
- We'll use the state tool to:
 - Install the dependencies and runtime environment required to run our project, and create a virtual environment for our existing project.

Real World: Project Setup

```
git clone https://github.com/ActiveState/tensorflask  
cd tensorflask  
state activate
```

Q & A

ActiveState®

Thank you to our panelist

Pete Garcin, Senior Product Manager, ActiveState (@rawktron)

What's Next

- Try the State Tool & ActiveState Platform
<https://platform.activestate.com>



Where to find us

Tel: **1.866.631.4581**

Website: **www.activestate.com**

Twitter: **@activestate**

Facebook: **/activestatesoftware**

