Improving the developer experience with Sitecore CLI

Jesper Balle & Sebastian Winslow, Valtech

SITECORE® SYMPOSIUM 2022

MEET EVERY MOMENT

October 17-20

#SitecoreSYM

Today's speakers

Jesper Balle

Tech Lead & Solution Architect, Valtech

- 18+ years of experience with Sitecore
- Sitecore Hackathon 2022 Winner

jesper.balle@valtech.com
@ballejesper
www.valtech.com

Sebastian Winslow

CTO, Denmark, Valtech

- 18+ years of experience with Sitecore
- Sitecore MVP 2009-2022
- Sitecore Hackathon 2021 Winner

sebsatian.winslow@valtech.com
@w1nsl0w
www.valtech.com

Agenda

01 Introduction (already done)

02 Sitecore CLI: The basics and key pointers

03 Example: Provisioning of XM environments

04 Example: Third-party services integration

05 Process automation

06 Summary

07 Q&A

MEET EVERY MOMENT

SITECORE®
SYMPOSIUM
2022

MEET
EVERY
MOMENT

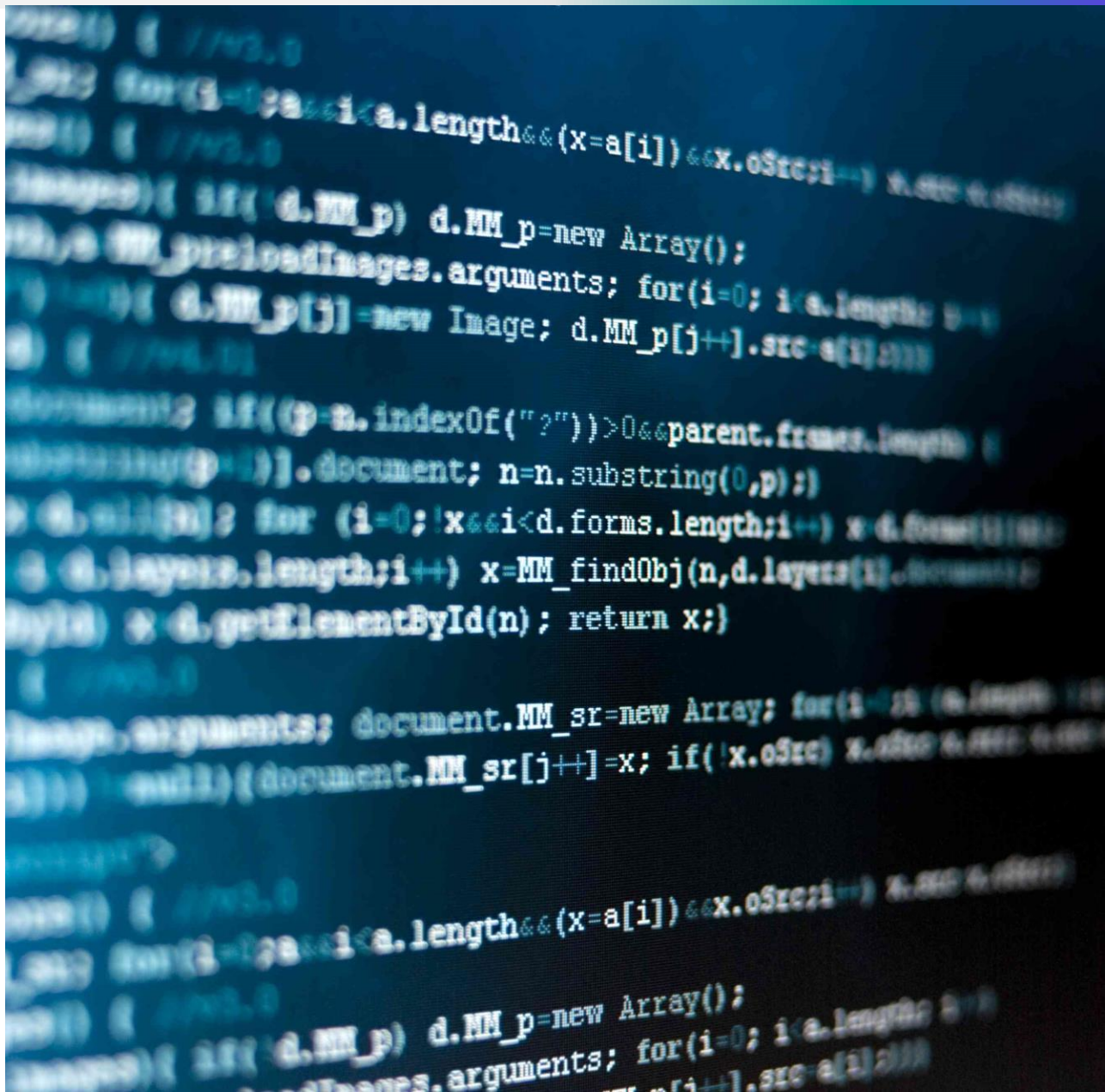
Sitecore CLI

Basics and key pointers

Why Sitecore CLI

Sitecore Command Line Interface (CLI) allows console communication with a Sitecore instance.

- Automation of processes
- Increased developer experience
- Increased efficiency

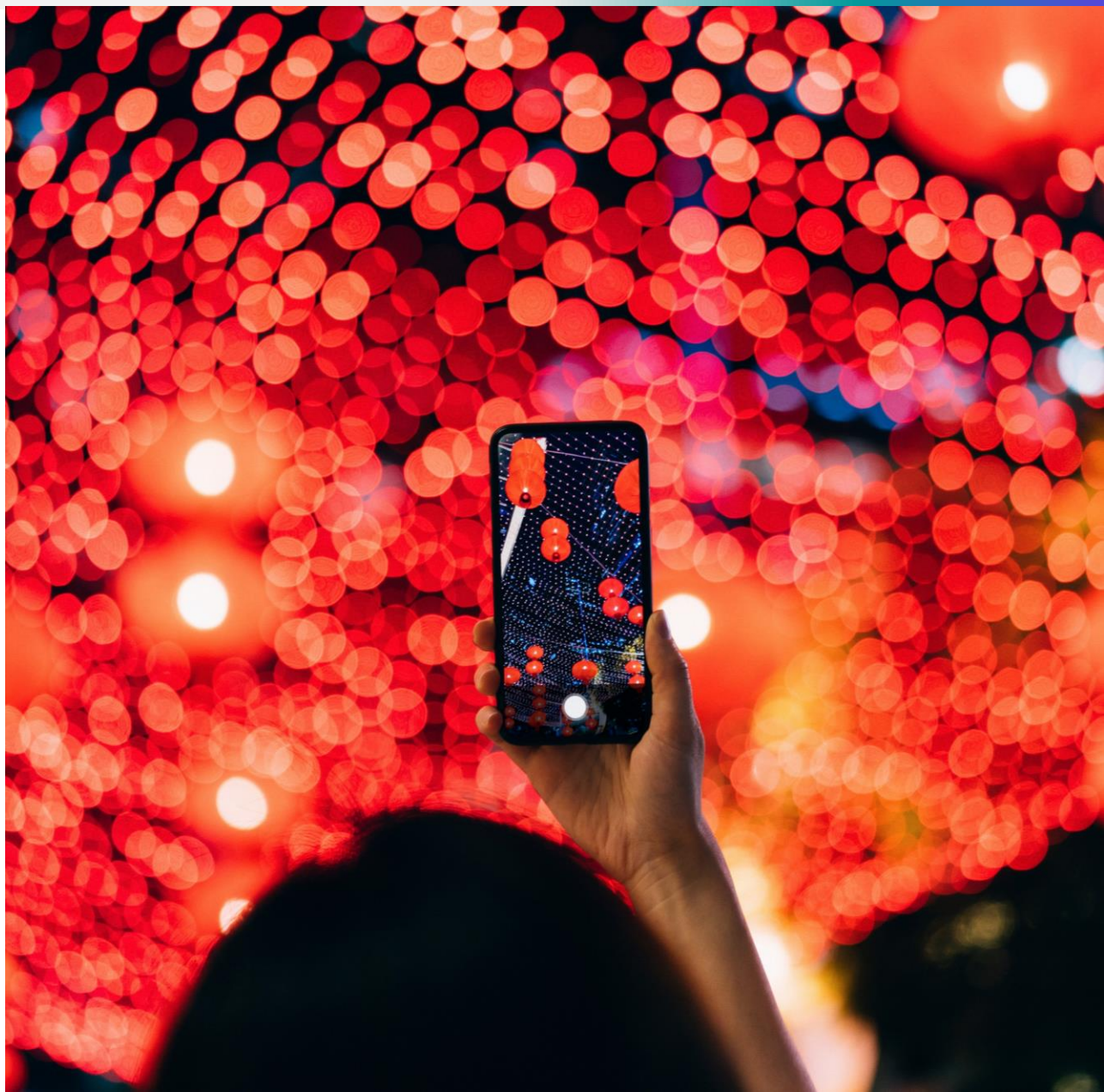


The Basics: Installation

- Install Sitecore CLI (Install Sitecore Command Line Interface)
  - Local vs. project scope
  - dotnet tool init / restore
  - dotnet sitecore plugin add
- Plugins provides functionality
- Several plugins are included when running dotnet sitecore init
- Example: Sitecore Content Serialization

The Basics: Building a plugin

- Plugins are distributed as NuGet packages
- Implement interface ISitecoreCliExtension
- Commands adhering to System.CommandLine
- Plugin assembly must be at plugin/{packageName}.dll in package



The Basics: Building a plugin


```
<PropertyGroup>
    <GeneratePackageOnBuild>true</GeneratePackageOnBuild>
    <PackageId>$(AssemblyName)</PackageId>
    <Title>Sitecore CLI extension for timeregistration with Clockify</Title>
    <Description>Nice package</Description>
    <VersionPrefix>1.0.0.2</VersionPrefix>


  <!--https://docs.microsoft.com/en-au/nuget/reference/msbuild-targets#including-content-in-a-package -->
  <PropertyGroup>
      <TargetsForTfmSpecificContentInPackage>$(TargetsForTfmSpecificContentInPackage);CustomBuildOutputTarget</TargetsForTfmSpecificContentInPackage>
  </PropertyGroup>
  <Target Name="CustomBuildOutputTarget">
      <ItemGroup>
          <TfmSpecificPackageFile Include="$(OutputPath)$(AssemblyName).dll">
              <PackagePath>plugin</PackagePath>
          </TfmSpecificPackageFile>
      </ItemGroup>
  </Target>
```

SITECORE®
SYMPOSIUM
2022

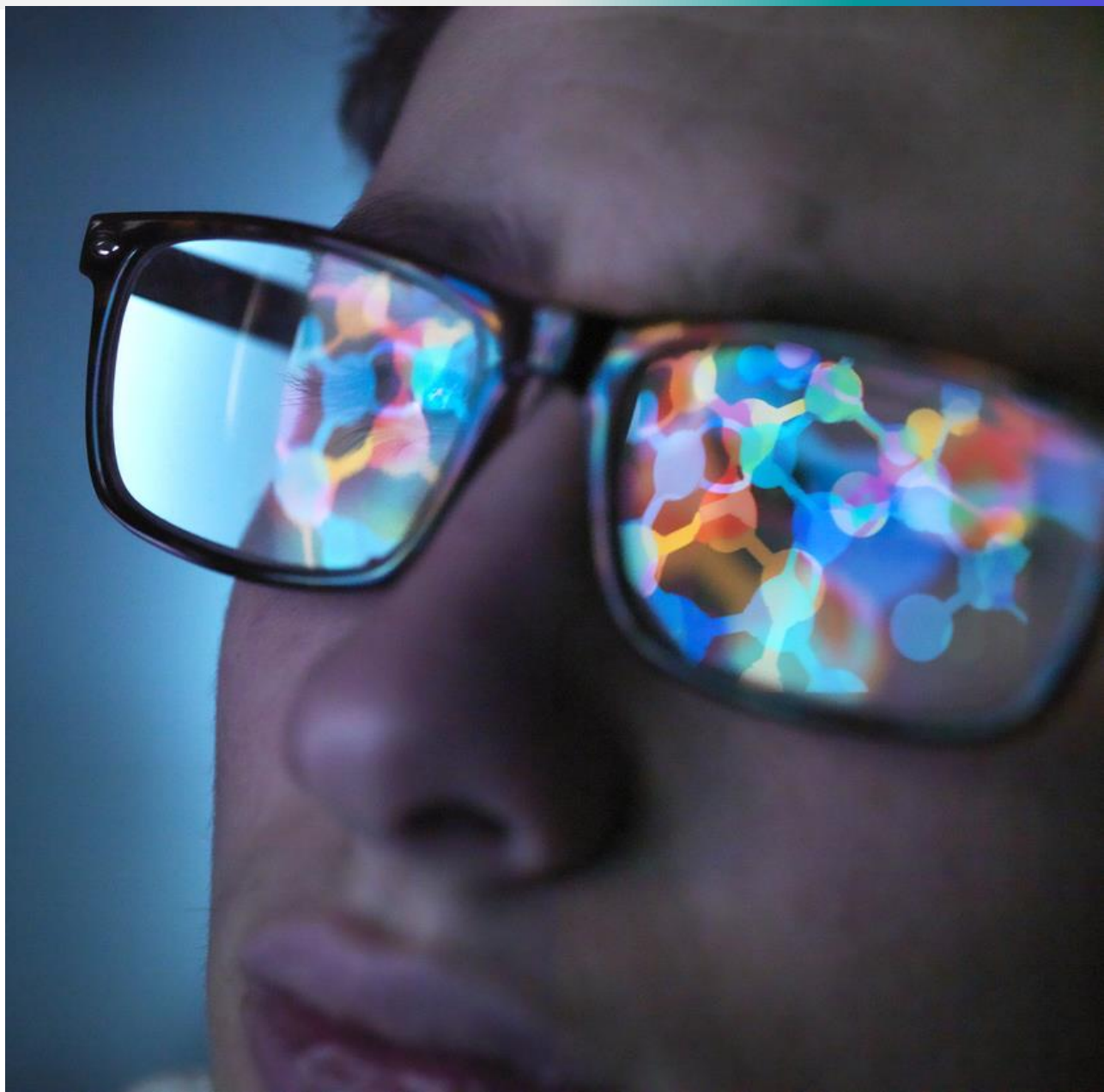
MEET
EVERY
MOMENT

Example 1:

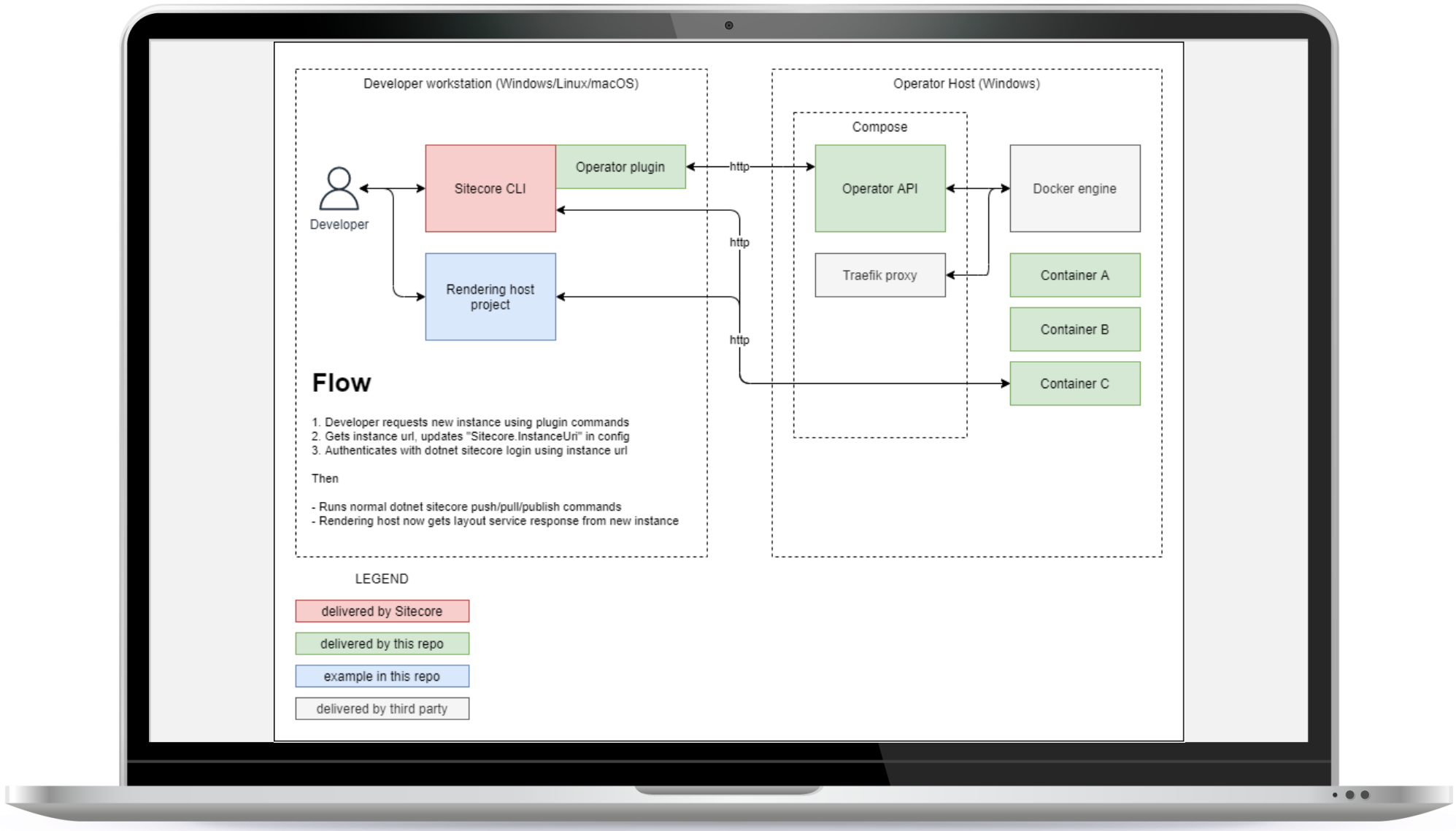
Provisioning XM instances

Purpose

- Support developing on arbitrary or low power devices, laptops, *nix.
- Headless development using layout service on a Sitecore XM instance.
- Still have independent environment per developer, proper versioning of templates, etc.
- One server with proper docker requirements and installation, developers don’t have to care about.



Architecture



In action!

```
PS C:\Projects\ValtechDK\Sitecore-Hackathon-2022-Web3\examples\rendering-host>
dotnet tool restore
Tool 'sitecore.cli' (version '4.1.0') was restored. Available commands: sitecor
e

Restore was successful.
PS C:\Projects\ValtechDK\Sitecore-Hackathon-2022-Web3\examples\rendering-host>
dotnet sitecore instance list
using http://operator-127-0-0-1.nip.io/ as operator!
Instances:
INSTANCENAME  STATE  URL
PS C:\Projects\ValtechDK\Sitecore-Hackathon-2022-Web3\examples\rendering-host>
dotnet sitecore instance start -n test --pwd b
Starting instance...
using http://operator-127-0-0-1.nip.io/ as operator!
Created http://test-127-0-0-1.nip.io
PS C:\Projects\ValtechDK\Sitecore-Hackathon-2022-Web3\examples\rendering-host>
$(invoke-webrequest http://test-127-0-0-1.nip.io/sitecore) | select -ExpandProp
erty StatusCode
200
PS C:\Projects\ValtechDK\Sitecore-Hackathon-2022-Web3\examples\rendering-host>
dotnet sitecore instance list
using http://operator-127-0-0-1.nip.io/ as operator!
Instances:
INSTANCENAME  STATE    URL
test          running  http://test-127-0-0-1.nip.io
PS C:\Projects\ValtechDK\Sitecore-Hackathon-2022-Web3\examples\rendering-host>
dotnet sitecore instance stop -n test
using http://operator-127-0-0-1.nip.io/ as operator!
Done
```

```
PS C:\Projects\ValtechDK\Sitecore-Hackathon-2022-Web3> .\Start.ps1
This project requires to map port 80 to a local traefik instance
We noticed that you have IIS running - as a precaution, we will stop IIS service (this requires
elevated mode)

VERBOSE: Performing the operation "Stop-Service" on target "World Wide Web Publishing
Service (w3svc)".
Stopping sitecore-web3_operator_1 ... done
Stopping sitecore-web3_traefik_1  ... done
Removing sitecore-web3_operator_1 ... done
Removing sitecore-web3_traefik_1  ... done
Network operatornet is external, skipping
Creating sitecore-web3_traefik_1  ... done
Creating sitecore-web3_operator_1 ... done
PS C:\Projects\ValtechDK\Sitecore-Hackathon-2022-Web3>
```

Time for actual code

```xml
<Project Sdk="Microsoft.NET.Sdk">
    <PropertyGroup>
        <SuppressDependenciesWhenPacking>true</SuppressDependenciesWhenPacking>
        <TargetFramework>netcoreapp3.1</TargetFramework>
        <TargetsForTfmSpecificContentInPackage>$(TargetsForTfmSpecificContentInPackage);CustomBuildOutputTarget</TargetsForTfmSpecificContentInPackage>
        <RootNamespace>Web3.Operator.Cli</RootNamespace>
        <GeneratePackageOnBuild>true</GeneratePackageOnBuild>
        <AssemblyName>Web3.Operator.Cli</AssemblyName>
        <PackageOutputPath>..\..\..\examples\rendering-host\nuget</PackageOutputPath>
        <NoWarn>NU5100;NU5128</NoWarn>
    </PropertyGroup>
    <ItemGroup>
      <None Remove=".gitignore" />
    </ItemGroup>

    <ItemGroup>
        <PackageReference Include="Sitecore.DevEx.Client.Cli" Version="4.1.1" />
    </ItemGroup>

    <Target Name="CustomBuildOutputTarget">
        <ItemGroup>
            <TfmSpecificPackageFile Include="$(OutputPath)Web3.Operator.Cli.dll" PackagePath="plugin" />
        </ItemGroup>
    </Target>
</Project>
```

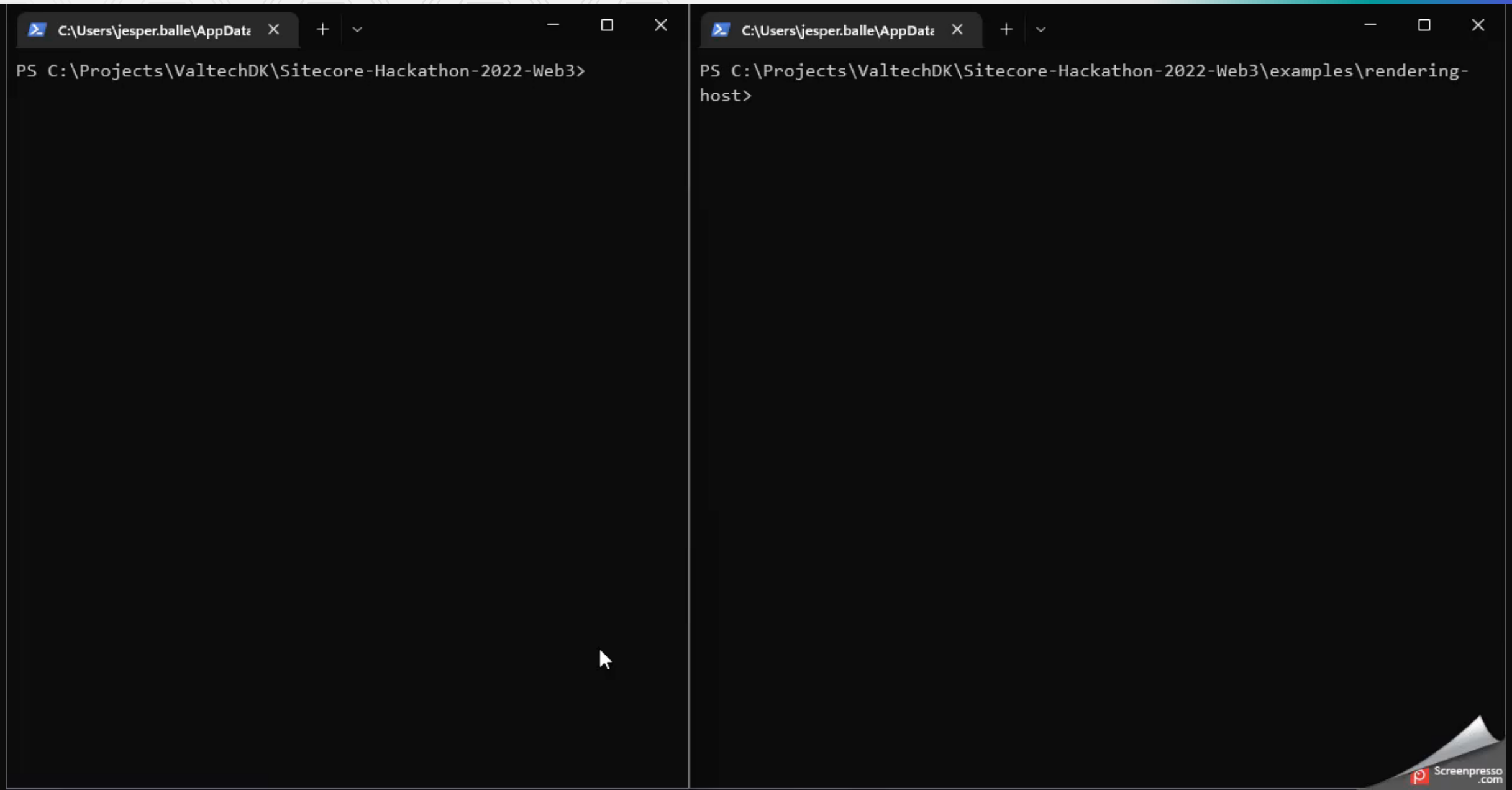
```
1  using Sitecore.Devex.Client.Cli.Extensibility.Subcommands;
2  using System;
3  using System.Threading.Tasks;
4  using Web3.Operator.Cli.Tasks;
5
6  namespace Web3.Operator.Cli.Commands
7  {
       3 references | Steven Aneel Hasz-Singh, 192 days ago | 1 author, 2 changes
8      public class StartInstanceCommand : SubcommandBase<StartInstanceTask, StartInstanceArgs>
9      {
           0 references | Steven Aneel Hasz-Singh, 192 days ago | 1 author, 2 changes
10         public StartInstanceCommand(IServiceProvider container) : base(name:"start", description:"Starts an instance", container)
11         {
12             AddAlias("nft");
13             AddOption(ArgOptions.SitecoreAdminPassword);
14             AddOption(ArgOptions.InstanceName);
15         }
16
           0 references | Steven Aneel Hasz-Singh, 192 days ago | 1 author, 1 change
17         protected override async Task<int> Handle(StartInstanceTask task, StartInstanceArgs args)
18         {
19             await task.Execute(args).ConfigureAwait(false);
20             return 0;
21         }
22     }
23 }
```

```
using Microsoft.Extensions.Logging;
using Sitecore.DevEx.Client.Logging;
using System.Threading.Tasks;
using Web3.Operator.Cli.Clients;
using Web3.Operator.Cli.Commands;

namespace Web3.Operator.Cli.Tasks
{
    6 references | Jesper Balle, 191 days ago | 2 authors, 5 changes
    public class StartInstanceTask
    {
        private readonly ILogger<StartInstanceTask> _logger;
        private readonly IOperatorClient _client;

        0 references | Jesper Balle, 192 days ago | 1 author, 1 change
        public StartInstanceTask(ILogger<StartInstanceTask> logger, IOperatorClient client)
        {
            _logger = logger;
            _client = client;
        }

        1 reference | Jesper Balle, 191 days ago | 2 authors, 5 changes
        public async Task Execute(StartInstanceArgs args)
        {
            args.Validate();

            _logger.LogConsoleInformation(message:"Starting instance...");
            var result = await _client.StartNewInstance(args.InstanceName, args.SitecoreAdminPassword);
            if(result != null)
            {
                _logger.LogConsoleInformation(message:$"{result.State} {result.Url}", System.ConsoleColor.Green);
            }
        }
    }
}
```

Future ideas and good to know

- HttpClientFactory is used – no external dependencies.
- To simplify a single image with both Sitecore and SQL server has been created.
- The need for Identity Server has been removed with a pipeline processor.
- Customizations for CM? An example could be mounting Azure File Share.
- https://github.com/Sitecore-Hackathon/2022-Web3

Process flows



Example 2:

Third-party integration

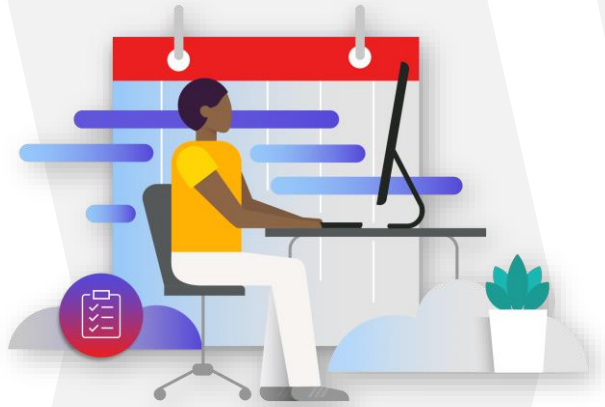
SITECORE® SYMPOSIUM 2022

MEET EVERY MOMENT

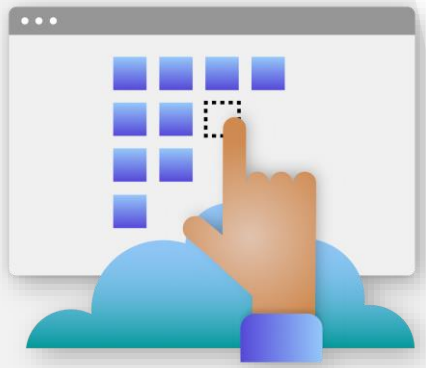
Task

- Common important (yet annoying) task: **time registration!**
- Let’s make this a simple and integrated part of the development workflow by using Sitecore CLI

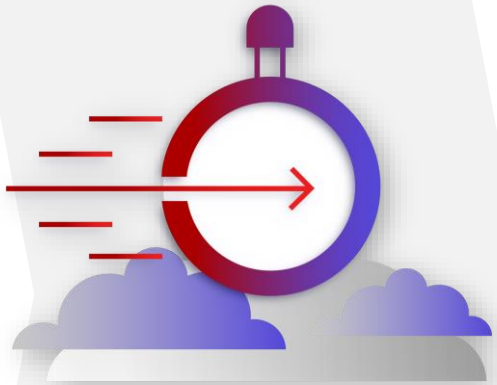
Flow



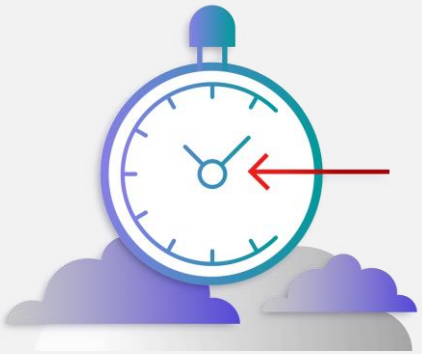
Get tasks



Select task



Start timer



Stop timer

Time for actual code

```
2 references | 0 changes | 0 authors, 0 changes
public class GetTasksTask : TaskBase<GetTasksArgs>
{
    private readonly ClockifyClient _client;

    0 references | 0 changes | 0 authors, 0 changes
    public GetTasksTask(ClockifyClient client)
    {
        _client = client;
    }

    2 references | 0 changes | 0 authors, 0 changes
    public override async Task<int> Execute(GetTasksArgs args)
    {
        var result :Response<List<…>> = await _client.FindAllTasksAsync(
            args.Workspace,
            args.Project,
            !args.IncludeInactive,
            args.Name, page:1, pageSize:100); // Task<Response<…>>

        if (!result.IsSuccessful || result.Data == null)
        {
            Console.Write(result.ErrorMessage);
            return -1;
        }

        if (!result.Data.Any())
        {
            Console.WriteLine("No task found");
            return -1;
        }

        var maxWidth :int = result.Data.Max(x :TaskDto => x.Id.Length);
        foreach (var item :TaskDto in result.Data)
        {
            Console.WriteLine($"{item.Id.PadRight(maxWidth)}  {item.Name}");
        }

        return 0;
    }
}
```

```
11  namespace SitecoreCliExtensions.Timereg.Clockify
12  {
        0 references | 0 changes | 0 authors, 0 changes
13      public class RegisterExtension : ISitecoreCliExtension
14      {
            0 references | 0 changes | 0 authors, 0 changes
15          public void AddConfiguration(IConfigurationBuilder configBuilder)
16          {
17          }
18
            0 references | 0 changes | 0 authors, 0 changes
19          public void AddServices(IServiceCollection serviceCollection)
20          {
21              serviceCollection.AddSingleton<UserConfigService>();
22              serviceCollection.AddSingleton<IUserConfigWriter, UserConfigService>();
23              serviceCollection.AddSingleton(sp :IServiceProvider => sp.GetRequiredService<UserConfigService>().GetSync());
24              serviceCollection.AddSingleton(ClockifyClientFactory.Create);
25          }
26
            0 references | 0 changes | 0 authors, 0 changes
27          public IEnumerable<ISubcommand> AddCommands(IServiceProvider container)
28          {
29              yield return new ClockifyTimelogCommand()
30                              .RegisterInitializeSubCommand(container)
31                              .RegisterGetTasksSubCommand(container); // ClockifyTimelogCommand
32
33          }
34      }
35  }
```

```
0 references | 0 changes | 0 authors, 0 changes
internal static class RegisterGetTasksCommandExtensions
{
    1 reference | 0 changes | 0 authors, 0 changes
    internal static ClockifyTimelogCommand RegisterGetTasksSubCommand(this ClockifyTimelogCommand rootCmd, IServiceProvider container)
    {
        var cmd = new GenericSubCommand<GetTasksTask, GetTasksArgs>(
            name:"tasks", description:"Fetch tasks", container);
        cmd.AddOption(ArgOptions.WorkspaceArg);
        cmd.AddOption(ArgOptions.ProjectArg);
        cmd.AddOption(ArgOptions.NameOption);

        rootCmd.AddCommand(cmd);
        return rootCmd;
    }
}
```

Include your dependencies

SITECORE SYMPOSIUM 2022

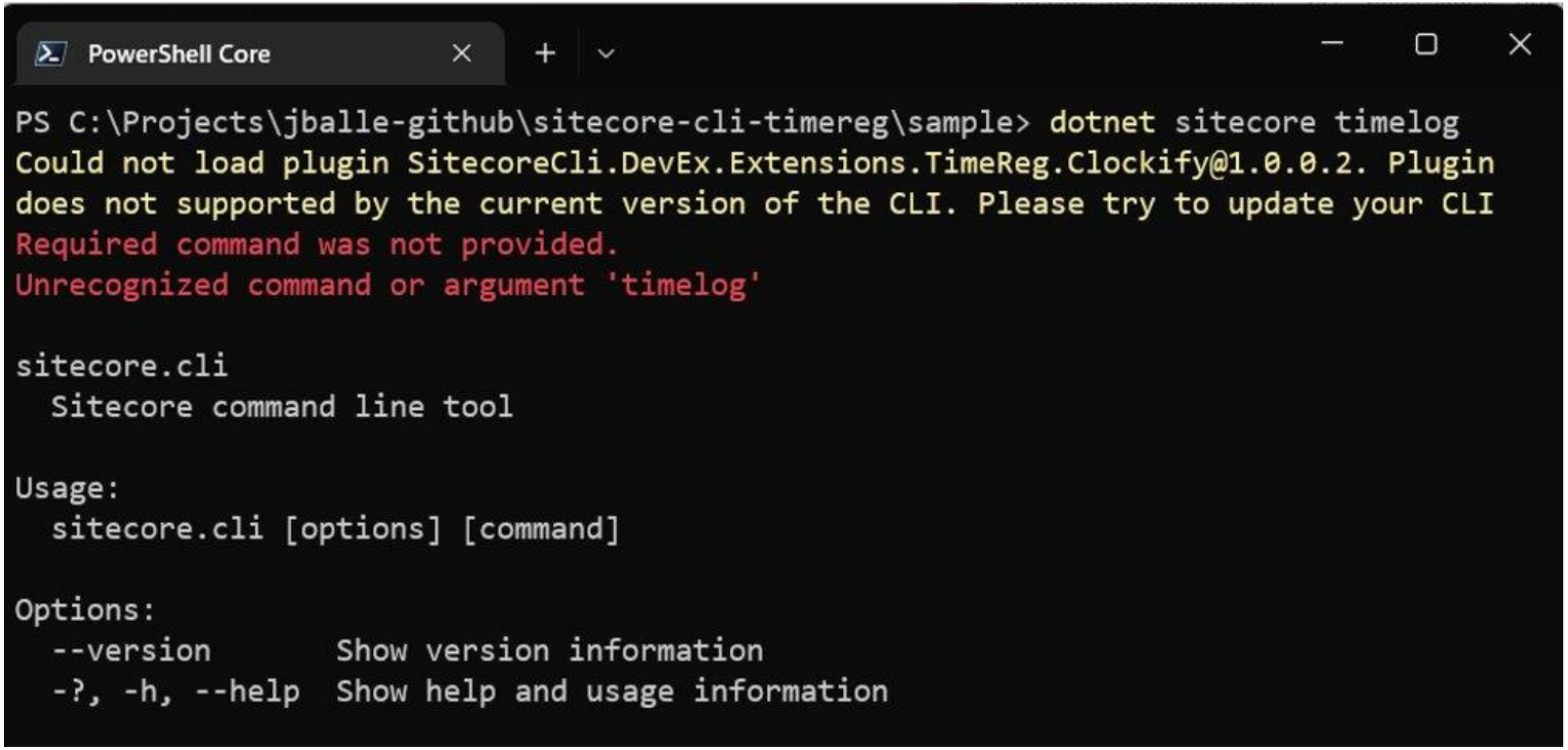
```
<ItemGroup>
    <PackageReference Include="Clockify.Net" Version="2.1.0">
    </PackageReference>
    <PackageReference Include="Newtonsoft.Json" Version="12.0.3" />
    <PackageReference Include="Sitecore.Devex.Client.Cli.Extensibility" Version="4.2.1">
        <PrivateAssets>all</PrivateAssets>
    </PackageReference>
</ItemGroup>

<PropertyGroup>
    <!-- forces SDK to copy dependencies into build output to make packing easier -->
    <CopyLocalLockFileAssemblies>true</CopyLocalLockFileAssemblies>
</PropertyGroup>

<Target Name="PackTaskDependencies" BeforeTargets="GenerateNuspec">
    <!--
    The include needs to happen after output has been copied to build output folder
    but before NuGet generates a nuspec. See https://github.com/NuGet/Home/issues/4704.
    -->
    <ItemGroup>
        <_PackageFiles Include="bin\$(Configuration)\$(TargetFramework)\Clockify*;bin\$(Configuration)\$(TargetFramework)\RestSharp*;bin\$(Configuration)
        <!--<_PackageFiles Include="@(ReferenceCopyLocalPaths)">-->
            <PackagePath>plugin</PackagePath>
            <Visible>false</Visible>
            <BuildAction>Content</BuildAction>
        </_PackageFiles>
    </ItemGroup>
</Target>
</Project>
```

Good to know

- CLI download and extract the NuGet package.
- PluginHelper in CLI uses AssemblyBindingContext.
- Assemblies must be available in the NuGet package!

Reflection loading error!

Sitecore default error message: “Plugin does not supported… Consider upgrading the CLI.”

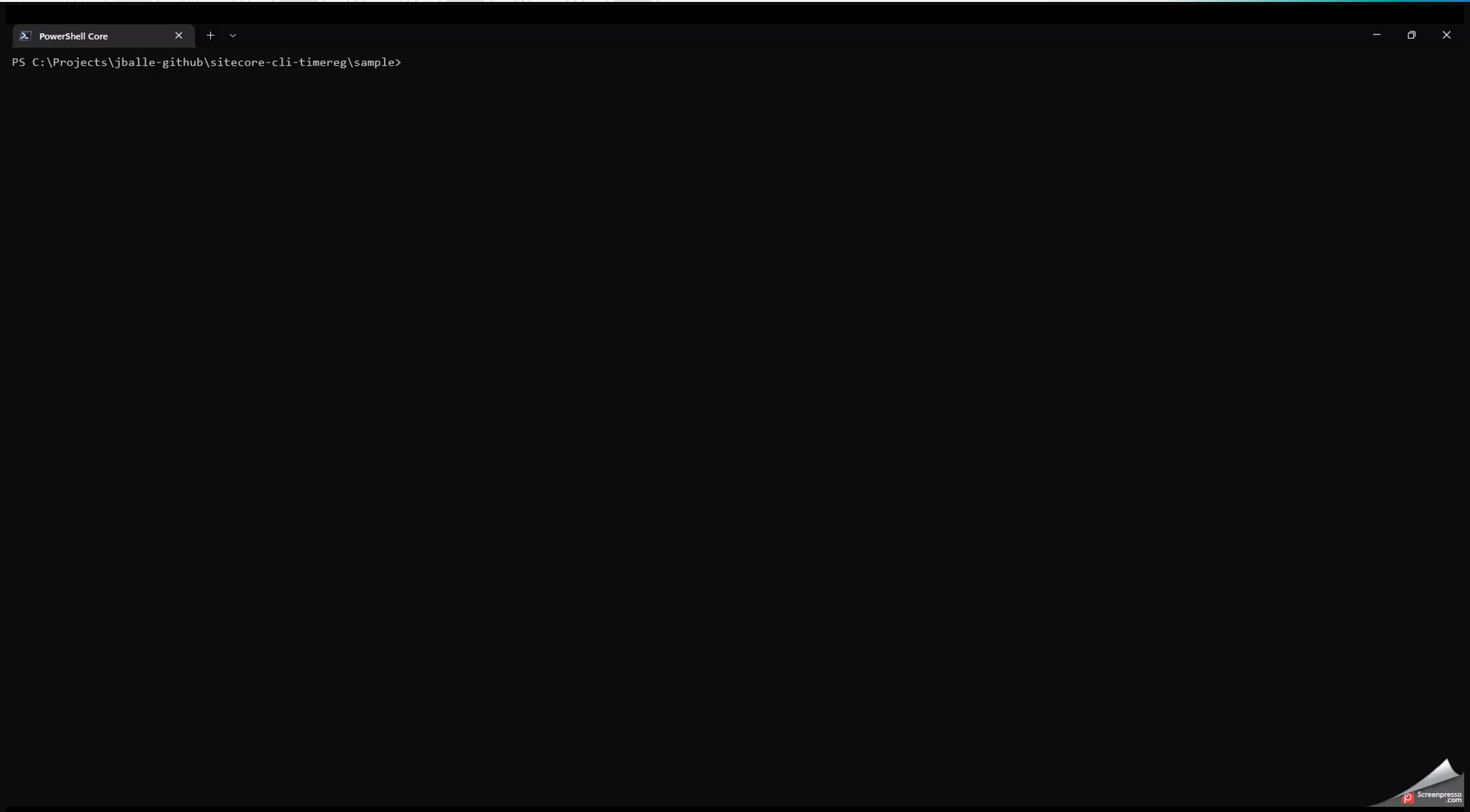


Reflection loading error revealed

```
var path = @"C:\Projects\jballe-github\Sitecore-cli-timereg\sample\.sitecore\package-cache\nuget\SitecoreCli.DevEx.Extensions.Timereg.Clockify.1.0.
try
{
    var ctx = new PluginLoadContext(path);
    var assembly = ctx.LoadFromAssemblyPath(path);
    var types :Type[] = assembly.GetTypes();
    Console.WriteLine("OK :)");
}
catch (System.Reflection.ReflectionTypeLoadE
{
    Console.WriteLine(ex.Message);
}
```

```
Microsoft Visual Studio Debu

Unable to load one or more of the requested types.
Could not load file or assembly 'RestSharp, Version=107.3.0.0, Culture=neutral, PublicKeyToken=59806
2e77f915f75'. The system cannot find the file specified.

C:\Projects\jballe-github\sitecore-cli-timereg\src\DebuggerApp\bin\Debug\net6.0\DebuggerApp.exe (pro
cess 51688) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatic
ally close the console when debugging stops.
Press any key to close this window . . .
```

In action



Process automation

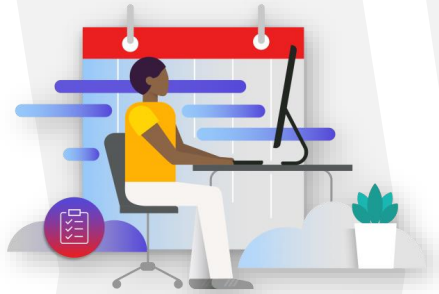
SITECORE®
SYMPOSIUM
2022

MEET
EVERY
MOMENT

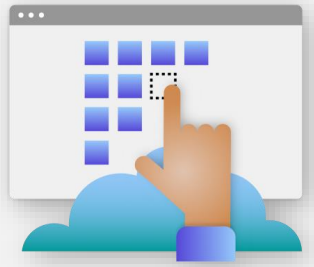
The bigger picture

- We really like CLI
- It is just a way to interact with a system
- What are common actions we do through the day that we would like to simplify/get rid of?

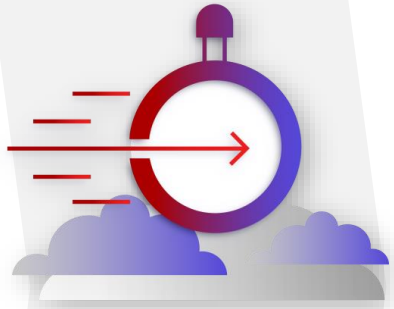
Flow



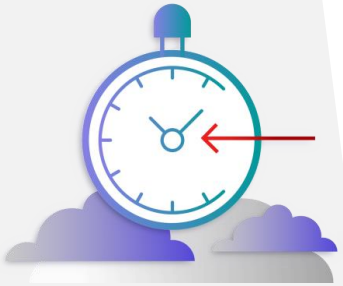
Get tasks from JIRA



Select task
(And optionally create branch)

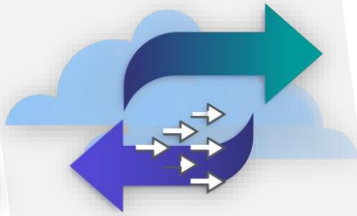


Start timer



Stop timer
(Optionally)

Commit
(Comment optional)



Commit & push
(Comment & Pull request optional)

Time for pseudo code

sample > .sitecore > {} workflows.json > ...

```json
{
  "variables": {
    "project": "symposium"
  },
  "workflows": [
    {
      "command": "start",
      "args": [
        {
          "name": "-n",
          "variableName": "taskName",
          "required": true
        }
      ],
      "actions": [
        {
          "cmd": "clockify-cli list --project ${project} --active --name ${taskName} --quiet",
          "outputVariable": "taskId"
        },
        { "cmd": "git checkout feature/${taskName}" },
        { "cmd": "git pull" },
        { "cmd": "clockify-cli in -p ${project} --task ${taskId}" },
        { "cmd": "dotnet sitecore ser watch" }
      ]
    },
    {
      "command": "stop",
      "actions": [
        { "cmd": "git commit -a" },
        { "cmd": "git push" },
        { "cmd": "clockify-cli out" }
      ]
    },
    {
      "command": "done",
      "actions": [
        { "cmd": "git commit -a" },
        { "cmd": "git push" },
        { "cmd": "clockify-cli out" }
      ]
    },
    {
      "command": "done",
      "actions": [
        { "cmd": "git commit -a" },
        { "cmd": "git merge main --rebase" },
        { "cmd": "git push" },
        { "cmd": "gh pr create --title ${taskName}" },
        { "cmd": "clockify-cli out" }
      ]
    }
  ]
}
```

Future ideas and good to know

- Configurable workflow/tasks
- The one shown is not complete, not in use and only to inspire!

Conclusions

SITECORE® SYMPOSIUM 2022

MEET EVERY MOMENT

Thoughts

- Sitecore CLI is a flexible tool available at your development hands.
- We can build new cool tasks.
- Since CLI and Open APIs in general is “a thing” you can quickly achieve very powerful results.
- Clockify just to exemplify – do any (timeregistration) plugin.

Q&A

SITECORE®
SYMPOSIUM
2022

MEET
EVERY
MOMENT

SITECORE®
SYMPOSIUM
2022

MEET
EVERY
MOMENT

Thank you

FOR DISCUSSION PURPOSES ONLY.

#SitecoreSYM