Condo

Build by Convention

bit.ly/condo-live

bit.ly/condo-deck

Deavon M. McCaffery

github.com/dmccaffery

github.com/automotiveMastermind

Condo

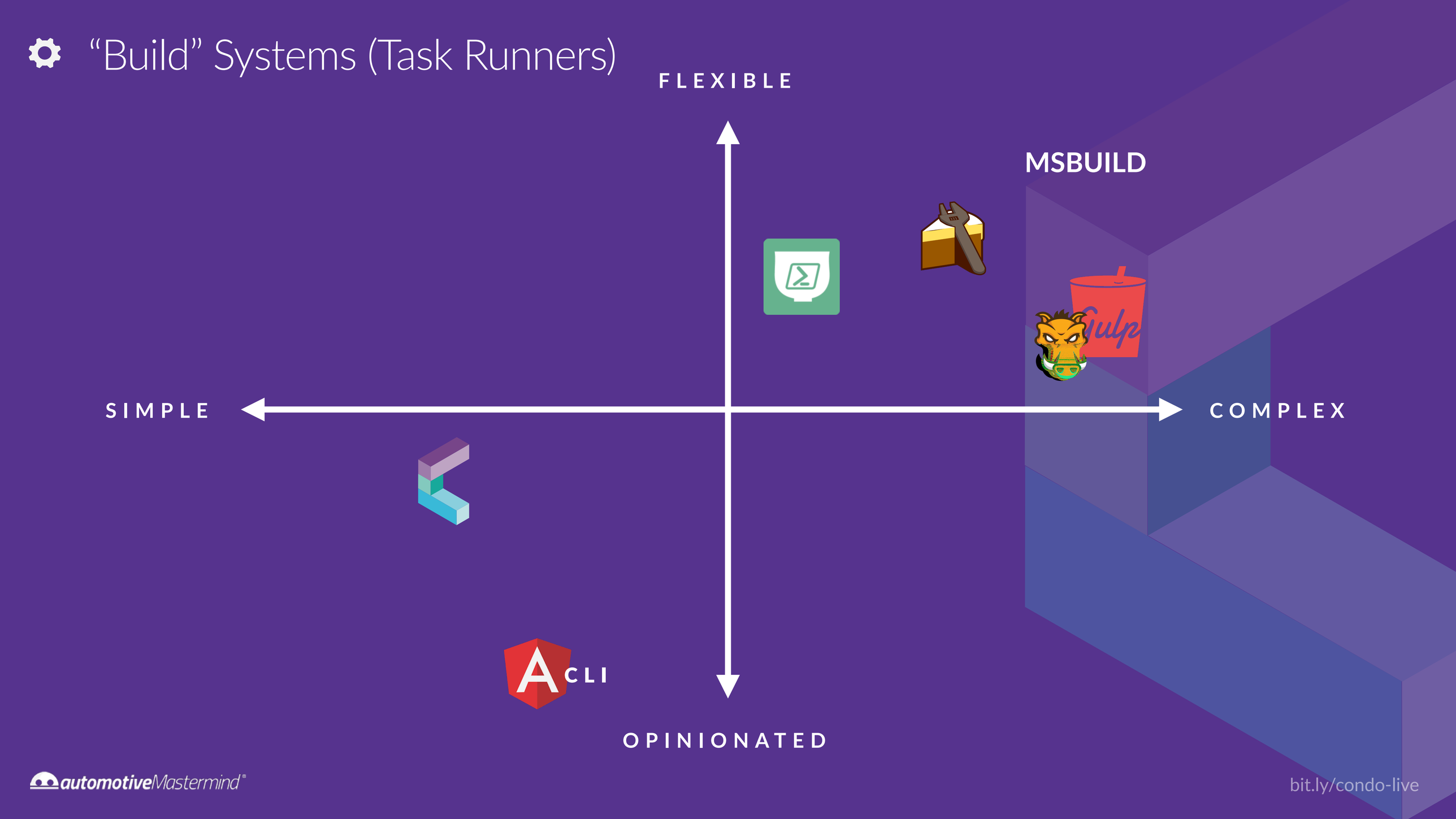
YET ANOTHER BUILD SYSTEM
SEE IT IN ACTION (DEMO)
THE BUILD LIFECYCLE

ESTABLISHED CONVENTIONS
CONFIGURATION
WHAT'S COMING NEXT

Yet Another Build System

Why Do We Want/Need This?!

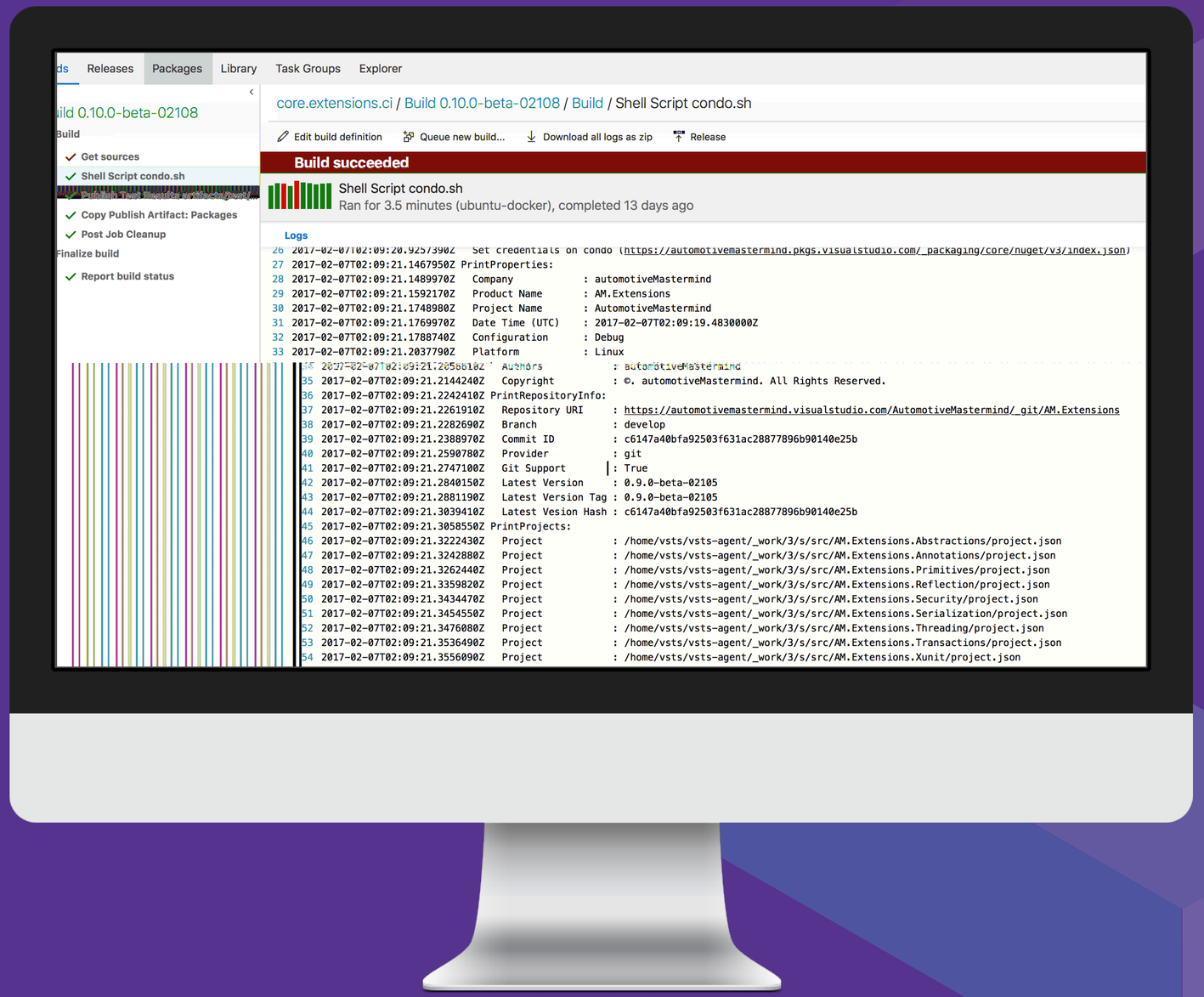
“Build” Systems (Task Runners)



Introduction to Condo

An open-source, conventional build system used to automate ***most*** build tasks without authoring complex scripts

- Automatic semantic versioning
- Automatic test discovery and execution
- Automatic change log generation
- Automatic packaging (NuGet, etc)
- Supports macOS, Linux, and Windows
- Works with .NET and .NET Core (among others)
- No Preinstalled Prerequisites Required
- Super Simple Configuration



See it in Action

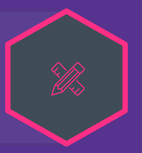
DEMO

The Build Lifecycle

Understanding the Build Pipeline

Condo Lifecycle


Initialize

Test


Bootstrap



Package


Version

Verify


Prepare



Document


Compile

Publish


Condo Lifecycle

Initialize

Get the Current Time (Local or NTP)
Get Platform Information (OS, etc)
Get Repository Information (Commit History)
Get Project Metadata

Print Build Configuration
Print Repository Information
Print Detected Projects
Print Build Information
Print Artifact Paths

Condo Lifecycle

Bootstrap 

Bootstrap Protected NuGet Feeds
Bootstrap Protected NPM Feeds

Bootstrap Dependencies
- NPM
- Gulp
- Grunt
- Bower
- DocFX

Condo Lifecycle

Version

Determine Build Quality
Determine Semantic Version

Update <VersionPrefix> in MSBuild Projects
Emit Assembly Info (Condo.AssemblyInfo.cs)

Condo Lifecycle

Prepare

Create Artifact Paths
Restore Git Submodules

Restore Packages
- NuGet
- NPM
- Bower

Condo Lifecycle

Compile

Compile .NET Projects
Compile .NET Core Projects

Execute Task Runners
- NPM Script
- Gulp
- Grunt
- Angular CLI

Condo Lifecycle

Test

Discover Tests
- VSTest
- xUnit
- NUnit
- MSTest

Execute Tests
- Platform Specific
- Agent Specific

NOTE: At present all test types will be executed (Unit, Integration, EndToEnd, and Performance)

Condo Lifecycle

Package NuGet
  .NET Core - csproj or project.json
  nuspec
Package NPM
  package.json
Package Bower
  bower.json

Package

Condo Lifecycle

NOTE: Currently has no default implementation, but can be used to perform tasks similar to the following:

Code Signing
Signature Verification
Package Verification

Verify

Condo Lifecycle

Generate Documentation
- DocFX (coming soon)
- GitHub Pages (coming soon)

Document

Condo Lifecycle

Publish Build Artifacts
  ./artifacts/build
Publish Test Artifacts
  ./artifacts/test
Publish Package Artifacts
  ./artifacts/package
Publish Documentation Artifacts
  ./artifacts/docs
Publish Feed
  ./artifacts/feed (local copy)
Publish Release Artifacts
  ./artifacts/publish

Publish

Established Conventions

Getting the Most from Condo

Folder Convention

src

Contains projects that contribute to deployable artifacts

test

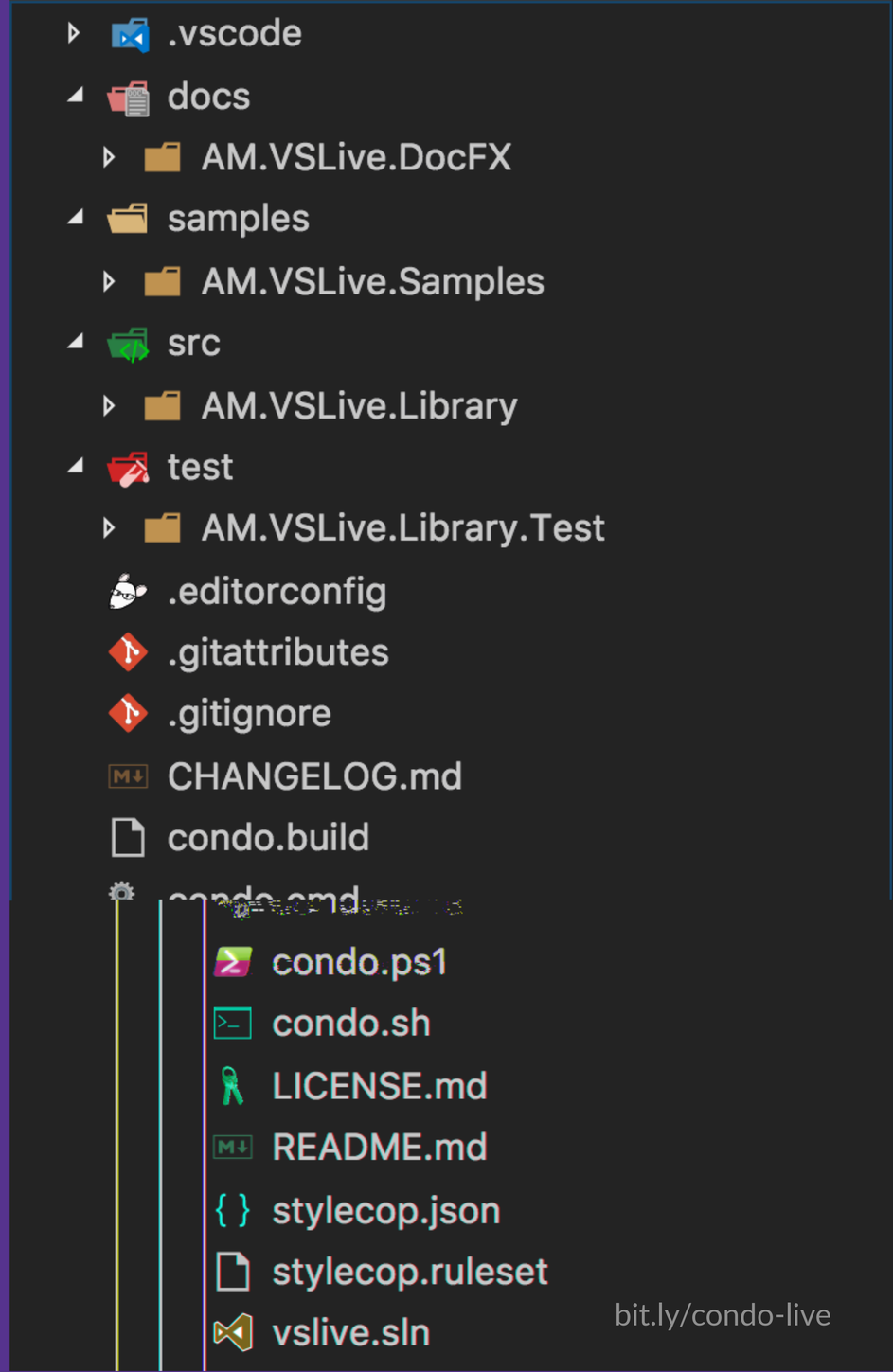
Contains projects that contribute to testing, including unit, integration end-to-end, and performance tests.

docs

Contains projects used to generate documentation, such as DocFX

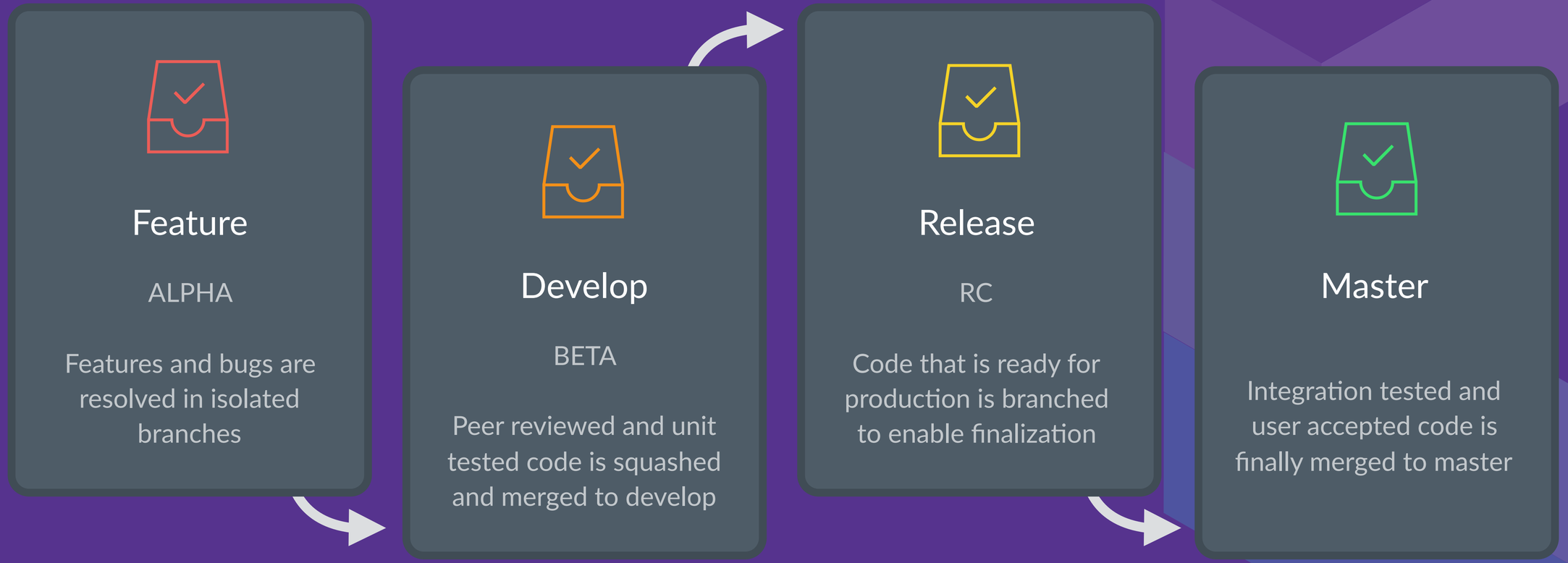
samples

Contains projects used to provide examples or code samples



bit.ly/condo-live

Branch Strategy Convention



Commit Message Convention

<type>(<scope>): <subject>
<BLANK LINE>
<body>
<BLANK LINE>
<footer>

<type>: a well-known, predefined value that represents the reason for the commit

<scope>: a predefined value (by project) that represents the area of concern

<subject>: a short, imperative sentence fragment in the present tense that describes the reason for the commit

<body>: an imperative, long form description of the reason for the commit

<footer>: used to identify breaking changes as well as work item references

Commit Message Convention

feat

Changes that introduce a new feature

fix

Changes that introduce a fix to an existing bug

build or ci

Changes that effect the build system or continuous integration configuration

test

Changes that add additional tests or corrects existing tests

refactor

Changes that refactors existing code without adding a new feature or fixing a bug

perf

Changes that improve performance

style

Changes that do not effect the meaning of the code (white space, formatting, missing semi-colons, etc)

docs

Changes that effect conceptual or generated documentation

Commit Message Convention

feat

Changes that introduce a new feature

fix

Changes that introduce a fix to an existing bug

SEMANTIC VERSIONING

build or ci

Changes that effect the build system or continuous integration configuration

test

Changes that add additional tests or corrects existing tests

refactor

Changes that refactors existing code without adding a new feature or fixing a bug

perf

Changes that improve performance

style

Changes that do not effect the meaning of the code (white space, formatting, missing semi-colons, etc)

docs

Changes that effect conceptual or generated documentation

Commit Message Convention

feat(transactions): simplify the invoker implementation

- remove embedded resource mappings as they are ignored by vs sdk
- simplify the invoker implementation

NOTE:

The .NET Core SDK preview tooling for Visual Studio 2015 ignores the buildOptions -> embed -> mappings when generating embedded resources, such as resx, which causes the namespace to change when building within Visual Studio. This code change removes those mappings and relies on the conventional namespace generated within Visual Studio.

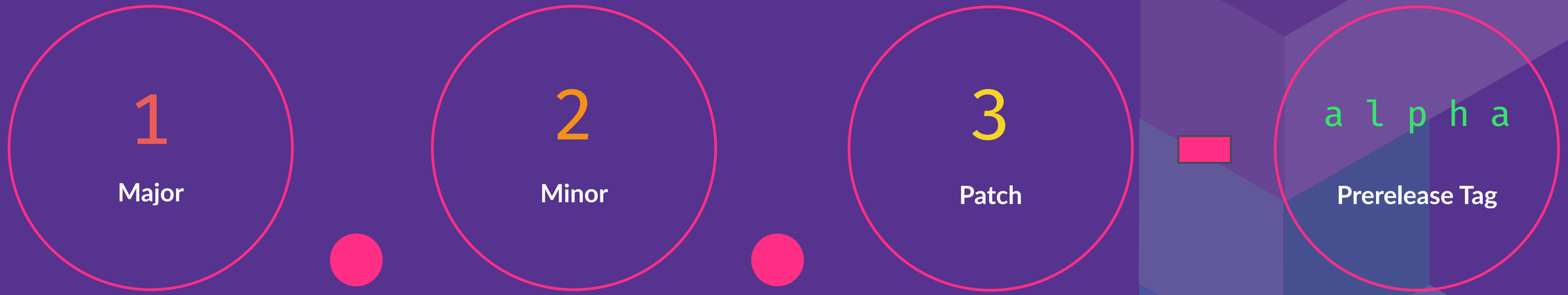
This commit also resolves the unnecessary use of a locally scoped variable within the ActivityScope class as pointed out by @keith during the last code review.

BREAKING CHANGE:

The signature of the ```aMm.Transactions.IInvoker<TRequest>``` interface has changed to ```aMm.Transactions.IInvoker<TProtocol, TRequest>```. This change has had a cascading effect on the ```IRetryPolicy.SendAsync``` method signature.

Related work items: #4656

Semantic Versioning



Changes that introduce a breaking change to public API

Identified by ‘BREAKING CHANGE’ in a commit message footer

Changes that introduce a new feature

Identified by a commit message with a type of ‘feat’

Changes that introduce a bug fix

Identified by a commit message with a type of ‘fix’

Changes in build quality

Identified by the git branch from which the build was initiated

Change Log Generation

> All notable changes to this project will be documented in this file.

0.1.0-beta-02075 (2017-01-31)

Features

- **collections:** add support for distinct 3b6dd16, closes #4908
- **di:** add infra, service accessor abstraction 7ad95c3, closes #4908
- **ipv6:** remove failing ipv6 cases 0c05843
- **transactions:** simplify the invoker implementation 9796236, closes #4656
- initial implementation of core extensions e5ea011, closes #4656

> BREAKING CHANGE

- **transactions:**

tocol,

- The signature of the `aMm.Transactions.IInvoker<TRequest>` interface has changed to `aMm.Transactions.IInvoker<TPro TRequest>`. This change has had a cascading effect on the `IRetryPolicy.SendAsync` method signature.

Configuration

Customizing the Build

Minimal Configuration

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="15.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003"
         InitialTargets="Clean" DefaultTargets="Build">
    <PropertyGroup>
        <Product>AM.Condo</Product>
        <StartDateUtc>2015-05-28</StartDateUtc>

        <Company>automotiveMastermind</Company>
        <Authors>automotiveMastermind</Authors>

        <License>MIT</License>
        <LicenseUri>https://opensource.org/licenses/MIT</LicenseUri>
    </PropertyGroup>

    <Import Project="$(CondoTargetsPath)\Lifecycle.targets" />
    <Import Project="$(CondoTargetsPath)\Goals.targets" />
</Project>
```

Branch Strategy Configuration

```
<PropertyGroup>
    <BranchStrategy>GitFlow  BranchStrategy>
</PropertyGroup>
```

Set the BranchStrategy property to one of the following:

- GitFlow
- GitLabFlow
- GitHubFlow

The default value is GitFlow (if not specified). This is also fully customizable regardless of the flow that is chosen.

Branch Strategy Configuration (GitFlow)

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="15.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
    <DevelopBranch  Condition=" '$(DevelopBranch)'  == '' ">develop</DevelopBranch>
    <MasterBranch   Condition=" '$(MasterBranch)'   == '' ">master</MasterBranch>

    <FeatureBranchPrefix  Condition=" '$(FeatureBranchPrefix)'  == '' ">feature</FeatureBranchPrefix>
    <BugfixBranchPrefix   Condition=" '$(BugfixBranchPrefix)'   == '' ">bugfix</BugfixBranchPrefix>
    <ReleaseBranchPrefix  Condition=" '$(ReleaseBranchPrefix)'  == '' ">release</ReleaseBranchPrefix>
    <SupportBranchPrefix  Condition=" '$(SupportBranchPrefix)'  == '' ">support</SupportBranchPrefix>
    <HotfixBranchPrefix   Condition=" '$(HotfixBranchPrefix)'   == '' ">hotfix</HotfixBranchPrefix>

    <DefaultBuildQuality        Condition=" '$(DefaultBuildQuality)'       == '' ">alpha</DefaultBuildQuality>

    <DevelopBranchBuildQuality  Condition=" '$(DevelopBranchBuildQuality)' == '' ">beta</DevelopBranchBuildQuality>
    <MasterBranchBuildQuality   Condition=" '$(MasterBranchBuildQuality)'  == '' "></MasterBranchBuildQuality>

    <FeatureBranchBuildQuality  Condition=" '$(FeatureBranchBuildQuality)' == '' ">$(DefaultBuildQuality)</FeatureBranchBuildQuality>
    <BugfixBranchBuildQuality
```

Version Strategy Configuration

```
<PropertyGroup>
    <VersionStrategy>Conventional</VersionStrategy>
</PropertyGroup>
```

Set the VersionStrategy property to one of the following:

- Conventional
- Manual

The default value is Conventional, which generates semantic versioning based on the BranchStrategy and ConventionStrategy configuration.

Manual will not generate the version or write the assembly info files.

Version Strategy Configuration

```
<PropertyGroup>
    <ConventionStrategy>Angular</ConventionStrategy>
</PropertyGroup>
```

Set the ConventionStrategy property to one of the following:

- Angular

The default value is Angular, which uses AngularJS style commit messages.

More convention strategies will be coming soon. In the meantime, all properties are fully customizable.

Version Strategy Configuration (Angular)

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="15.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
    <FieldPattern           Condition=" '$(FieldPattern)'           == '' ">^-(.*?)-$</FieldPattern>

    <HeaderPattern          Condition=" '$(HeaderPattern)'          == '' ">^(\w*)(?:\(([\w\$\.\-\* ]*)\))?\: (.*)$</HeaderPattern>
    <HeaderCorrespondence   Condition=" '$(HeaderCorrespondence)'   == '' ">Type;Scope;Subject</HeaderCorrespondence>

    <RevertPattern          Condition=" '$(RevertPattern)'          == '' ">^Revert\s""([\s\S]*)""\s*This reverts commit (\w*)\.</RevertPattern>
    <RevertCorrespondence   Condition=" '$(RevertCorrespondence)'   == '' ">RevertHeader;RevertHash</RevertCorrespondence>

    <MergePattern           Condition=" '$(MergePattern)'           == '' "></MergePattern>
    <MergeCorrespondence    Condition=" '$(MergeCorrespondence)'    == '' "></MergeCorrespondence>

    <ActionKeywords         Condition=" '$(ActionKeywords)'         == '' ">Close;Closes;Closed;Fix;Fixed;Resolve;Resolves;Resolved</ActionKeywords>
    <NoteKeywords           Condition=" '$(NoteKeywords)'           == '' ">BREAKING CHANGE;BREAKING CHANGES</NoteKeywords>

    <ReferencePrefixes      Condition=" '$(ReferencePrefixes)'      == '' ">#</ReferencePrefixes>
    <MentionPrefixes        Condition=" '$(MentionPrefixes)'        == '' ">@</MentionPrefixes>

    <IncludeInvalidCommits  Condition=" '$(IncludeInvalidCommits)'  == '' ">False</IncludeInvalidCommits>

    <ReleaseMessage         Condition=" '$(ReleaseMessage)'         == '' ">chore(release): </ReleaseMessage>

    <HeaderPartial          Condition=" '$(HeaderPartial)'          == '' ">$(MSBuildThisFileDirectory)$(slash)header.hbs</HeaderPartial>
    <FooterPartial          Condition=" '$(FooterPartial)'          == '' ">$(MSBuildThisFileDirectory)$(slash)footer.hbs</FooterPartial>
    <CommitPartial          Condition=" '$(CommitPartial)'          == '' ">$(MSBuildThisFileDirectory)$(slash)commit.hbs</CommitPartial>
    <ChangeLogTemplate      Condition=" '$(ChangeLogTemplate)'      == '' ">$(MSBuildThisFileDirectory)$(slash)template.hbs</ChangeLogTemplate>

    <GroupByHeader          Condition=" '$(GroupByHeader)'          == '' ">Type</GroupByHeader>
    <SortByHeader           Condition=" '$(SortByHeader)'           == '' ">Scope</SortByHeader>

    <MinorCorrespondence    Condition=" '$(MinorCorrespondence)'    == '' ">Type</MinorCorrespondence>
    <MinorValue             Condition=" '$(MinorValue)'             == '' ">feat</MinorValue>

    <ChangeLogTypes         Condition=" '$(ChangeLogTypes)'         == '' ">feat;fix;perf;revert</ChangeLogTypes>
    <ChangeLogNames         Condition=" '$(ChangeLogNames)'         == '' ">Features;Bug Fixes;Performance Improvements;Reverts</ChangeLogNames>
  </PropertyGroup>

  <ItemGroup>
    <ChangeLogPartials Include="$(HeaderPartial)" Condition=" Exists('$(HeaderPartial)') " />
    <ChangeLogPartials Include="$(FooterPartial)" Condition=" Exists('$(FooterPartial)') " />
    <ChangeLogPartials Include="$(CommitPartial)" Condition=" Exists('$(CommitPartial)') " />
  </ItemGroup>
</Project>
```

What's Coming Next

Coming Soon to a Repo Near You

What's Coming Next

- DocFX support on Windows or Mono
- GitHub Pages support on all platforms
- Additional OOB commit message conventions
- PoSH Module publishing for PoSH 5.0
- PoSH Module testing via Pester
- Up-to-date documentation on github.io
- Condo CLI for an improved initialization experience
- Configuration templates and user defaults