Quick Tizen Infrastructure Setup for Small Team

Jian-feng Ding, Alexander Kanevskiy
Open Source Technology Center, Intel
2013-11-11
Agenda

- The Problem
- What do we have currently in Tizen.org
- What we can do for one developer?
- What we can do for small teams?
- What we can do for not so small teams?
The Problem:
How to make Custom device using Tizen?

Making Tizen based Product
The Problem

- **In order to build custom device, vendor should be able to do**
  - Add custom applications
  - Add custom APIs
  - Tweak and modify some APIs
  - Use device specific HW adaptation layer
  - Re-build whole stack of software for specific platform

- **All of this require development environment where changes can be introduced easily and builds with modified software produced, deployed and tested rapidly**
What do we have currently @ Tizen.org
Tizen 3.0: Code flow in infrastructure

**git project path**
- platform/core/*
- platform/kernel/*
- platform/upstream/*
- platform/framework/*
- apps/*

**git branches**
- Tizen:IVI
- Tizen:Mobile

**OBS projects**
- ivi/
- mobile/

**Download URL**
- ivi/
- mobile/

**Legend:**
- Common packages
- Profile specific
- Branch/project/Folder

---

platform/core/*
platform/kernel/*
platform/upstream/*
platform/framework/*
apps/*
... ...

profile/mobile/*
Topology of all Tizen backend services

- Source
- WebUI/API
- Scheduler
- Repo
- Workers
- Gerrit
- OBS
- Jenkins and jobs
- Static files hosting
- BOSS
- Rabbitmq
- Redis
- BOSS daemons
- Image workers
- Data and Files
- Trigger events
- Control and uploading
Tizen Backend Infrastructure:
Designed to be scalable for 1000s of active developers

- **Gerrit**
  - Git Source Code Hosting service, Review board
- **Jenkins**
  - Continuous Integration engine
- **OBS**
  - Open Build System, extensible system to build packages for various packaging formats
- **BOSS**
  - Legacy process workflow engine.
- **Smaller building blocks and services**
  - Drupal
  - LDAP
  - MySQL
  - Redis
  - RabbitMQ
  - Xen
  - KVM
Tizen Backend Infrastructure – replication and deployment

- Multiple services in multiple servers with multiple platforms support.
- Lots of configurations and maintenance work required.
- Plenty of resources required for build node and image creation node.
- Something good for a huge team is not necessarily good for small teams
- Most of Tizen community members would need smaller and a lot simpler solutions for quick deployment and easy ramp-up
What we can do for one developer?
Client side build without infrastructure services
GBS and MIC

• **Two client side CLI tools**
  - GBS: “git build system”, which build out rpm files based on source code in git trees, and helper functions to support Tizen development workflow
  - MIC: “Mic the Image Creator”, the tools to create images in several formats

• **Can support most of the popular Linux distributions, depends on developers’ choice**
  - openSUSE 12.1, 12.2, 12.3
  - Fedora 18, 19
  - Ubuntu 12.04, 12.10, 13.04 (13.10 will come soon)
  - CentOS 6.x
GBS local build

• The major feature of GBS tools is the “local build” by using command line “gbs build”

• Fruitful command line options to meet most of the development requirements

• Based on the source code in developer’s machine
  • It can build on one single Git tree
  • And can also build multiple ones, in the proper order according build-require relationship

• For the build environment bootstrapping
  • GBS will use a genuine Tizen environment for building
  • For bootstrapping, GBS can use the rpm files from
    1 Tizen repositories in download.tizen.org
    2 Pre-built binary Git tree in tizen.org Gerrit for minimal set of Tizen packages
GBS local full build

• Allow user to download all the source code, build and create the image locally without interaction with remote network, just like Android development.

• Three steps:
  1. Clone all the git trees by using ‘repo’ command
  2. Build all the projects in local machine, to generate all the rpm files in local repo
  3. The ‘buildimage’ subcommand will call MIC to create the images, using the output files of above steps
Critical Git trees for Local-full-build

- `/scm/manifest.git`
  - The input of `repo` command
  - Different entry file for different profile
    1. `/scm/manifest/ivi.xml`
    2. `/scm/manifest/mobile.xml`
  - All the blow git trees are included in manifest also

- `/scm/meta/build-config.git`
  - Build config meta file for local build environment initialization

- `pre-built/toolchain-{arch}`
  - Smallest set of pre-built binary rpm files for build bootstrapping for all packages
  - Multiple ones for multiple architectures: ia32 and arm

- `/scm/meta/gbs-config.git`
  - Pre-defined configuration for GBS tool, which will tell GBS how to use the content in above meta git trees
What we can do for small teams?
Simple centralized build environment and Continuous Integration for small teams
Simple CI for small teams

- **Centralized file storage to store and serve build artifacts**
  - Generic Linux server with enough disk space and Apache
- **Centralized Git server(s)**
  - There are several ways of consolidating your project source code.
  - Examples could be
    1. plain Git repositories in user directories
    2. Managed git server via gitolite/gitosis
    3. Local powerful Git servers like Gerrit
    4. Git as a Service: GitHub, BitBucket, Gitorious
- **Continuous Integration engine**
  - Jenkins
  - Buildbot
  - TeamCity
Scaling local-full-build for small team

• GBS local full build is very helpful for one developer, however it can be utilized for more if you pair it with CI engines
  • Build can be triggered based on changes and/or time
  • Developers can pass parameters for builds in CI UI
  • Builds can be reproduced in clean environment
  • Results of builds are published to centralized storage and can be used by other developers
Our Recipe: “GBS plus Jenkins”

- **Ingredients:**
  - Gerrit (optional)
  - Jenkins
  - Build hosts with GBS+MIC installed as Jenkins workers

- Easy to deploy, easy to configure, and easy to maintain
- Jenkins allow to trigger builds on different events (10s of plugins) including manual, cron-like and change based
- Gerrit as optional component can provide very convenient way of triggering builds based on new patches, merges, tags
- Jenkins has great abilities to integrate with automatic testing systems
Topology of “GBS plus Jenkins”
What we can do for not so small teams?
Scalable CI system for not so small teams
When do you need more complex setup?

- Your team grows
- You need more complex release criteria than just merging code
- You are working on multiple different branches of the same code
- You want to have code “development” branches that automatically would be following master code line
- You want to rebuild only changed packages and their dependencies
- You want to secure and isolated environment for software you are building
Recipe for not so small teams

• Additional ingredients compared to previous setup
  • Gerrit as version control, review board and Jenkins build trigger
  • OBS for granular and distributed package build
  • Shared between Jenkins and OBS worker build nodes
  • Bootstrapped toolchain inside OBS
Recipe: how to setup all of those?

• **Host OS – OpenSUSE 12.3**
  • There is nothing special in installation. Use Minimal setup and create sysadmin accounts for yourself
  • On workers, create “jenkins” user and setup ssh key authorization between main node and workers

• **Additional repositories**
  • OBS: [http://download.opensuse.org/repositories/OBS:/Server:/2.4/openSUSE_12.3/](http://download.opensuse.org/repositories/OBS:/Server:/2.4/openSUSE_12.3/)
  • Jenkins services: [http://download.tizen.org/services/archive/13.08/openSUSE_12.3/](http://download.tizen.org/services/archive/13.08/openSUSE_12.3/)
Recipe: installation

- **Packages to install**
  - apache memcached
  - jenkins jenkins-plugins jenkins-jobs-common obs-event-plugin
  - obs-api obs-server build-initvm-i586
  - obs-source_service obs-service-gbs

- **Gerrit installation**
  - [https://gerrit-releases.storage.googleapis.com/gerrit-2.7.war](https://gerrit-releases.storage.googleapis.com/gerrit-2.7.war)
  - java -jar gerrit.war init -d /srv/gerrit/review_site

- **Configuring Apache**
  - /etc/apache2/vhosts.d/obs.conf

- **Configuring OBS**
  - /srv/www/obs/api/config/database.yml
  - /srv/www/obs/api/config/options.yml
  - /srv/www/obs/webui/config/database.yml
  - /srv/www/obs/webui/config/options.yml
  - /usr/lib/obs/server/BSConfig.pm
Recipe: bootstrapping OBS and Jenkins

- **OBS bootstrap**
  - Create Project for pre-built binaries (toolchain)
  - Create Your main project
  - Copy build configs into those projects from Tizen.org release
  - Copy binary RPM packages from Tizen.org release into project for pre-buils
  - Use “obs_admin –rescan-repository” to tell OBS to re-index imported binaries
  - Set your main project to be built on top of pre-built binaries

- **In Jenkins configuration UI**
  - Enable jobs: submit, image-creator, create-snapshot, mail-sender
  - Enable service jobs: obs-event-dispatcher, load-repos-config

- **Copy from Tizen.org and adjust according to your needs**
  - scm/meta/snapshot-repo-conf.git
Recipe: preparing source code

• **Easiest way of importing source code**
  • Use “repo” tool and manifest from desired Tizen.org release to download needed source code
  • Use “repo forall” to upload each git repository into your instance of Gerrit

• **Triggering first build**
  • You can use same approach of “repo forall” to do “gbs submit…” and “git push” to trigger submission of all source code OBS
  • After first build is done and in your main project you have initial set of built binaries, you can remove link between your main project and pre-builds.

• **Now you have standalone instance of Tizen build infrastructure 😊**
Things left behind the scenes and future plans

- **Behind the scenes**
  - In this simple example we didn’t discuss specifics of user authentication or access controls
  - SSH authentication between nodes
  - Download server setup
  - Rsync for uploading build results to download server

- **Future plans**
  - We planning to document that recipe with all details on Tizen.org
  - We are looking on selecting tools for automating deployment. It is possible that we will be providing in near future some set of scripts to easily deploy described recipe.
  - There is ideas about providing pre-installed and pre-configured minimally VM appliances to even further simplify setup for small and medium sized teams
Links

- **Tizen development workflow:**
  - [https://source.tizen.org/documentation/work-flow](https://source.tizen.org/documentation/work-flow)

- **Manual of GBS and MIC**
  - [https://source.tizen.org/documentation/reference/git-build-system](https://source.tizen.org/documentation/reference/git-build-system)
  - [https://source.tizen.org/documentation/articles/mic-image-creator](https://source.tizen.org/documentation/articles/mic-image-creator)

- **GBS Local full build user guide:**