Distributed Systems

Distributed Development of a Distributed System

Prof. Dr. Oliver Hahm

Frankfurt University of Applied Sciences
Faculty 2: Computer Science and Engineering
oliver.hahm@fb2.fra-uas.de
https://teaching.dahahm.de

16.04.2024

Agenda

- Case Study: RIOT
  - Internet of Things
  - RIOT
- Distributed Software Development
  - Distributed Software Development
  - Tooling
  - Git

Agenda

Agenda

The Internet of Things

What is the Internet of Things?

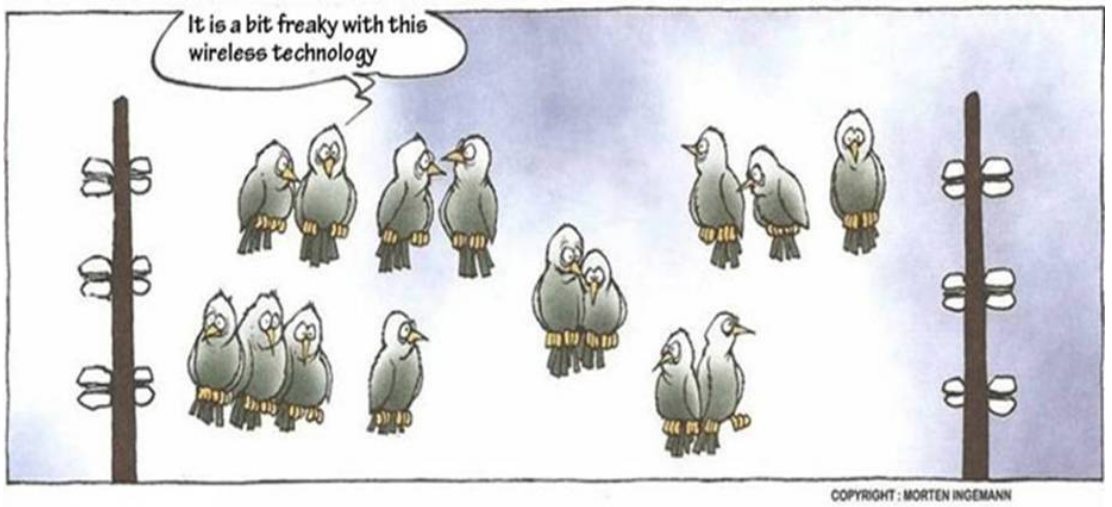
The Evolution of the IoT

Three Disruptive Technologies as the Roots of the IoT

The Evolution of the IoT

Three Disruptive Technologies as the Roots of the IoT

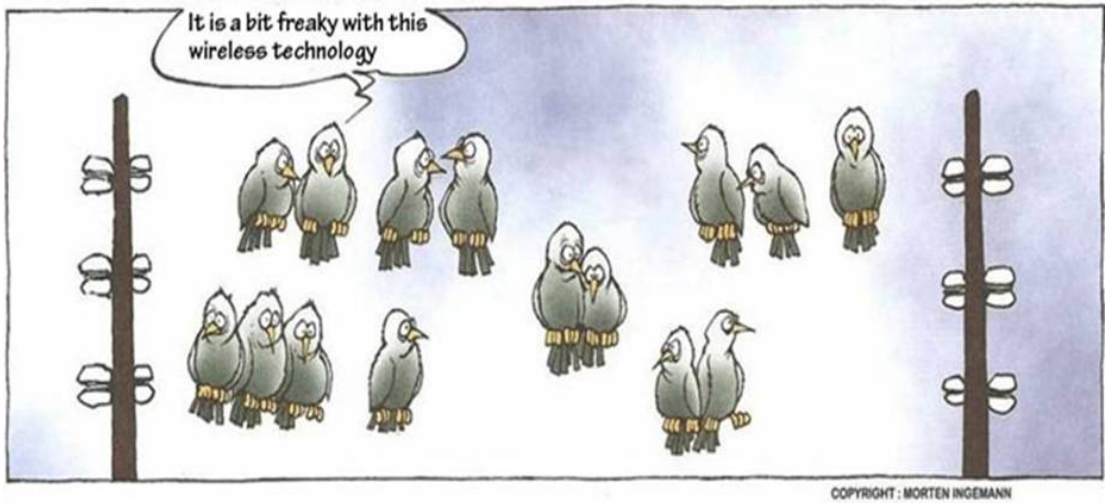
- Wireless Communication

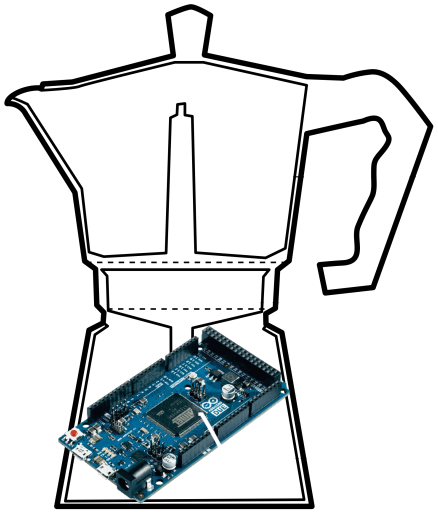


The Evolution of the IoT

Three Disruptive Technologies as the Roots of the IoT

- Wireless Communication
- Low-cost Embedded Systems

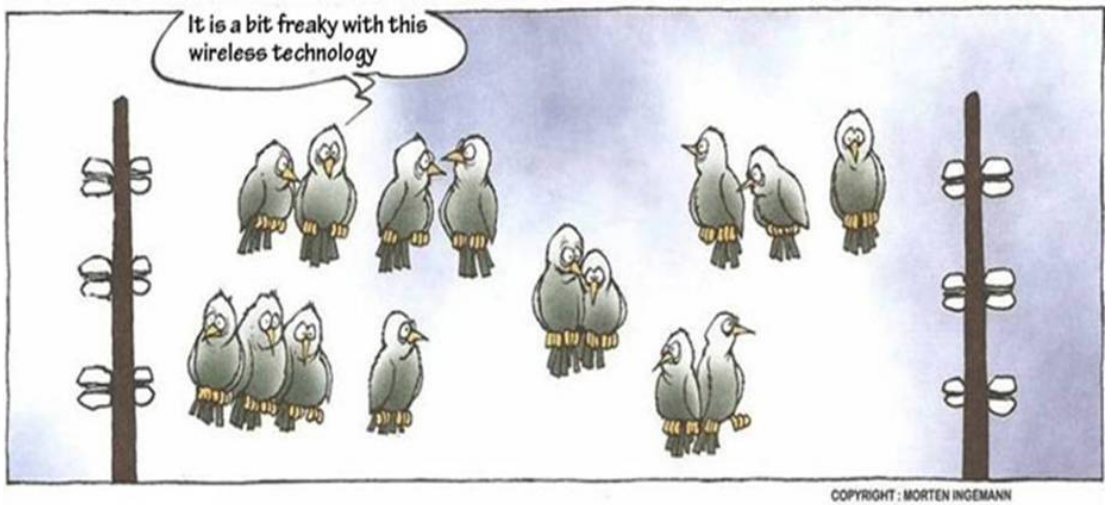


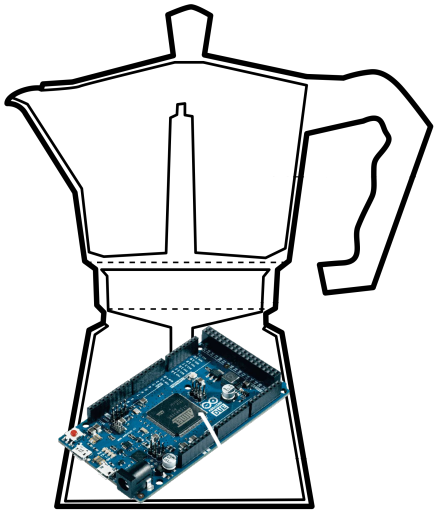


The Evolution of the IoT

Three Disruptive Technologies as the Roots of the IoT

- Wireless Communication
- Low-cost Embedded Systems
- The Internet





Smart Object Networking at Internet-Scale

Connecting the Physical World with the Internet

- Transforming Things into Smart Objects
- Enabling Interconnected Smart Services

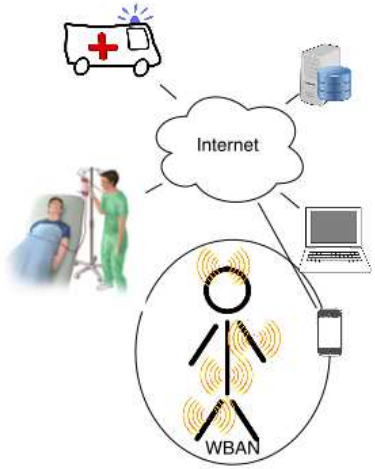
Smart Object Networking at Internet-Scale

Industrial Automation

Connecting the Physical World with the Internet

- Transforming Things into Smart Objects
- Enabling Interconnected Smart Services

Mobile Health



Micro & Nano Satellites

Building & Home Automation

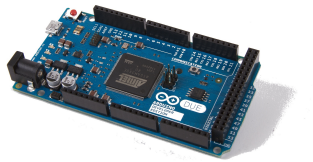
Challenges

Low-end IoT Devices: Limited Resources (RFC7228)

iotlab-m3

Senslab WSN430

Arduino Due



- Memory < 1 Mb
- CPU < 100 MHz
- Energy < 10 Wh

Requirements

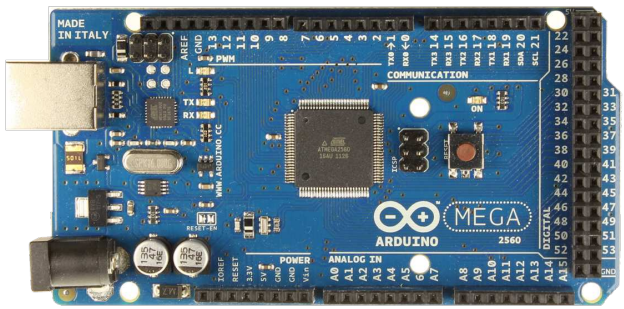
- Interoperability
- Energy Efficiency
- Reliability
- Latency
- Low Cost Factor
- Autonomy
- Security
- Scalability

Agenda

RIOT – The friendly OS for the IoT

"If your IoT device cannot run Linux, then use RIOT!"

- RIOT requires only a few kB of RAM/ROM, and a small CPU
- With RIOT, code once & run heterogeneous IoT hardware
  - 8bit hardware (e.g. Arduino)
  - 16bit hardware (e.g. MSP430)
  - 32bit hardware (e.g. ARM Cortex-M, x86)



IoT and Distributed Systems

Which properties and challenges of a Distributed System do apply here?

RIOT as a Distributed System

- RIOT provides a platform for IoT applications
- As such it faces many challenges of a distributed system
  - Network stacks for interoperability and message passing
  - Rock-solid architecture and seamless autoconfiguration to ensure robustness and autonomy
  - Support for usable and efficient cryptographic operations to enable security
  - Synchronization of processes and states across multiple nodes in a network
  - Provide access to, for example, sensor data independent from its location

RIOT as a Distributed Software Project

- RIOT developers are distributed all over the world
- This poses various challenges to the software development process
  - Maintaining a common code base
  - Testing a huge variety of configurations on a plethora of supported hardware
  - Keeping track of bugs and security concerns
  - Alignment on common goals and collaboration on work packages

Agenda

- Case Study: RIOT
  - Internet of Things
  - RIOT
- Distributed Software Development
  - Distributed Software Development
  - Tooling
  - Git

Agenda

- Case Study: RIOT
  - Internet of Things
  - RIOT
- Distributed Software Development
  - Distributed Software Development
  - Tooling
  - Git

How do you share code in a team?

What problems needs to be addressed?

Challenges in Software Development

Challenges in Software Development

- Distribute software among developers

Challenges in Software Development

- Distribute software among developers
- Track and log changes

Challenges in Software Development

- Distribute software among developers
- Track and log changes
- Identify contributors

Challenges in Software Development

- Distribute software among developers
- Track and log changes
- Identify contributors
- Perform code reviews

Challenges in Software Development

- Distribute software among developers
- Track and log changes
- Identify contributors
- Perform code reviews
- Work on multiple features in parallel

Challenges in Software Development

- Distribute software among developers
- Track and log changes
- Identify contributors
- Perform code reviews
- Work on multiple features in parallel
- Enable roll-backs to an older state

Challenges in Software Development

- Distribute software among developers
- Track and log changes
- Identify contributors
- Perform code reviews
- Work on multiple features in parallel
- Enable roll-backs to an older state
- Mark a certain state of the software $\rightarrow$ versioning

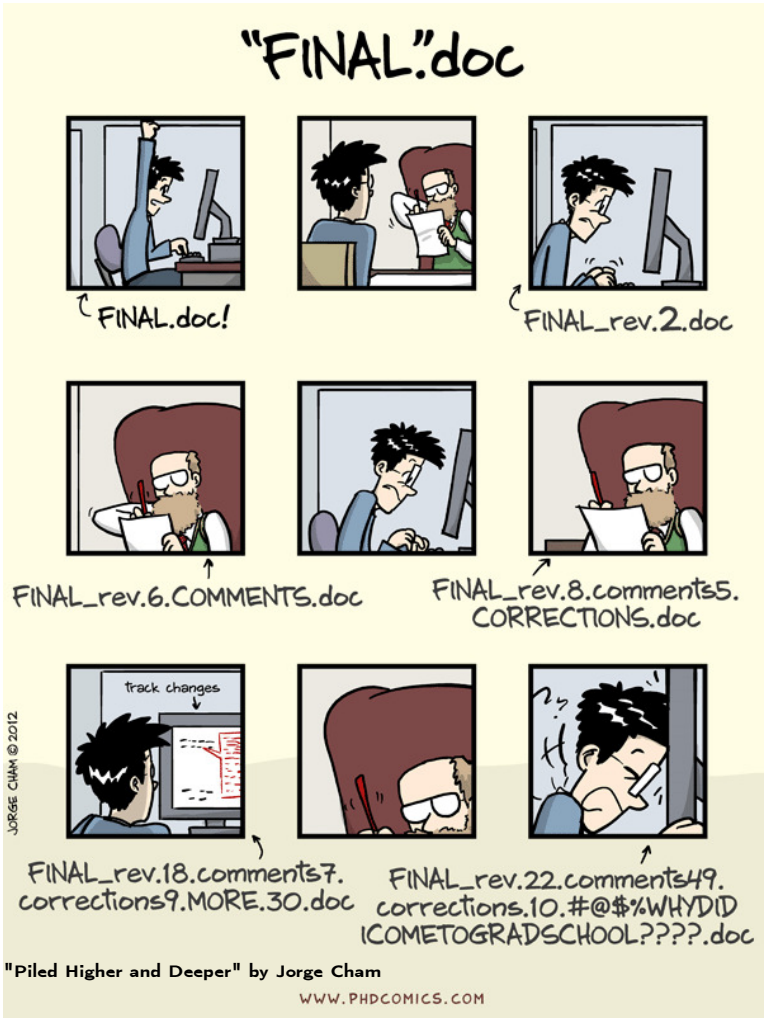
Agenda

Version Control Systems (VCS)

- Used for:
  - Code
  - Documentation
  - Configuration
  - Collaboration

Other Names

- Source Control Management (SCM)
- Version Control Software
- Revision Control Software



www.phdcomics.com

Problems to be solved by a VCS

Solves a lot of problems

- *I have changes I want to integrate (merge) into the main project.*
- *I want to track the state of this project over time.*
- *I want to make some changes without possibly breaking what I have.*

Answers questions like . . .

- *Who wrote this module?*
- *When was this particular line of this particular file edited? By whom? Why was it edited?*
- *Over the last 1000 revisions, when/why did a particular unit test stop working?*

Benefits of VCS

- VCS provides numerous benefits for both working environment

Benefits of VCS

- VCS provides numerous benefits for both working environment
- Individual benefits
  - Backups
  - Tagging – marking the particular version in time
  - Branching – multiple versions
  - Tracking changes
  - Revert (undo) changes (Rollback)

Benefits of VCS

- VCS provides numerous benefits for both working environment
- Individual benefits
  - Backups
  - Tagging – marking the particular version in time
  - Branching – multiple versions
  - Tracking changes
  - Revert (undo) changes (Rollback)
- Team benefits
  - Working on the same code base in a (distributed) team
  - Merging concurrent changes
  - Support for conflicts resolution when the same file (the same part of the file) has been simultaneously changed by several developers
  - Determine the author and time of the changes

Common VCS Tools

Frequently Used VCS Tools

- Concurrent Versions System (CVS)
- Subversion (SVN)
- ClearCase
- Git
- GNU Bazaar
- Mercurial
- Azure DevOps

Centralized VCS

There are two main types of VCS: distributed and centralized. They each have their pros and cons.

Centralized VCS

- Single repository
- Version (revision) IDs are usually sequential numbers
- Centralized Version Control Systems require a persistent connection with a centralized server
  $\Rightarrow$ Straightforward authentication, but single point of failure
- Examples include:
  - Subversion
  - CVS

Distributed VCS

Distributed Version Control Systems (DVCS)

- *"I'm going to work over here for a while and tell you about what I did later."*
- Every user has a full copy of the repository
  $\Rightarrow$ Offline work usually possible
- Version IDs are usually a GUID (Globally Unique IDentifier) or hash value
- Examples include:
  - Git ($\rightarrow$ Github or Gitlab)
  - Mercurial

Terminology of VCS

The following list contains some concepts or terms you often encounter when working with VCS. Some systems use slightly different verbiage to describe these ideas but you can get by with these words.

- Repository
  Contains your project.
  "I created a new repository (repo) for my school project so we can collaborate more easily."
- Diff
  The delta (additions and deletions) between two states of a project.
  "The diff between draft one and two was very long thanks to the help of the skilled editor."
- Revision
  A version of a file or the entire repository.

Terminology of VCS (2)

- **Commit**
  A snapshot of your project's state at a point in history. Records the difference between two points in history with a diff.
  "Your last commit modified two methods and introduced a bug somewhere."
- **Revert**
  Roll back a commit from the repository.
- **Branch**
  Modifications to a project (main branch = trunk) made in parallel with the a main branch, but not affecting the main branch.
  "My branch introduces some changes which might break production so I'm not going to merge it until it's well tested."

Terminology of VCS (3)

- **Merge**
  Introducing changes from one branch into another.
  "I merged three commits into the Master branch so we can have those features in the next release."
- **Conflict**
  When a file cannot be merged cleanly (automagicaly).
- **Fork**
  Your own version of somebody else's project where you take the original code-base and make modifications. May include many changes or just a few bug-fixes. Sometimes you end up merging those changes back upstream.
- **Clone**
  Downloading a local copy of a project.

Agenda

"git can mean anything"

Linus Torvalds

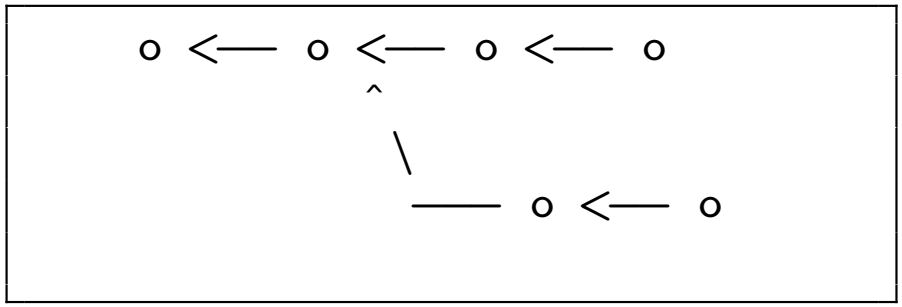
"I'm an egoistical bastard, and I name all my projects after myself. First 'Linux', now 'Git' "

"Git is MacGyver,
Mercurial is James Bond."

https://importantshock.wordpress.com/2008/08/07/git-vs-mercurial/

The Data Model

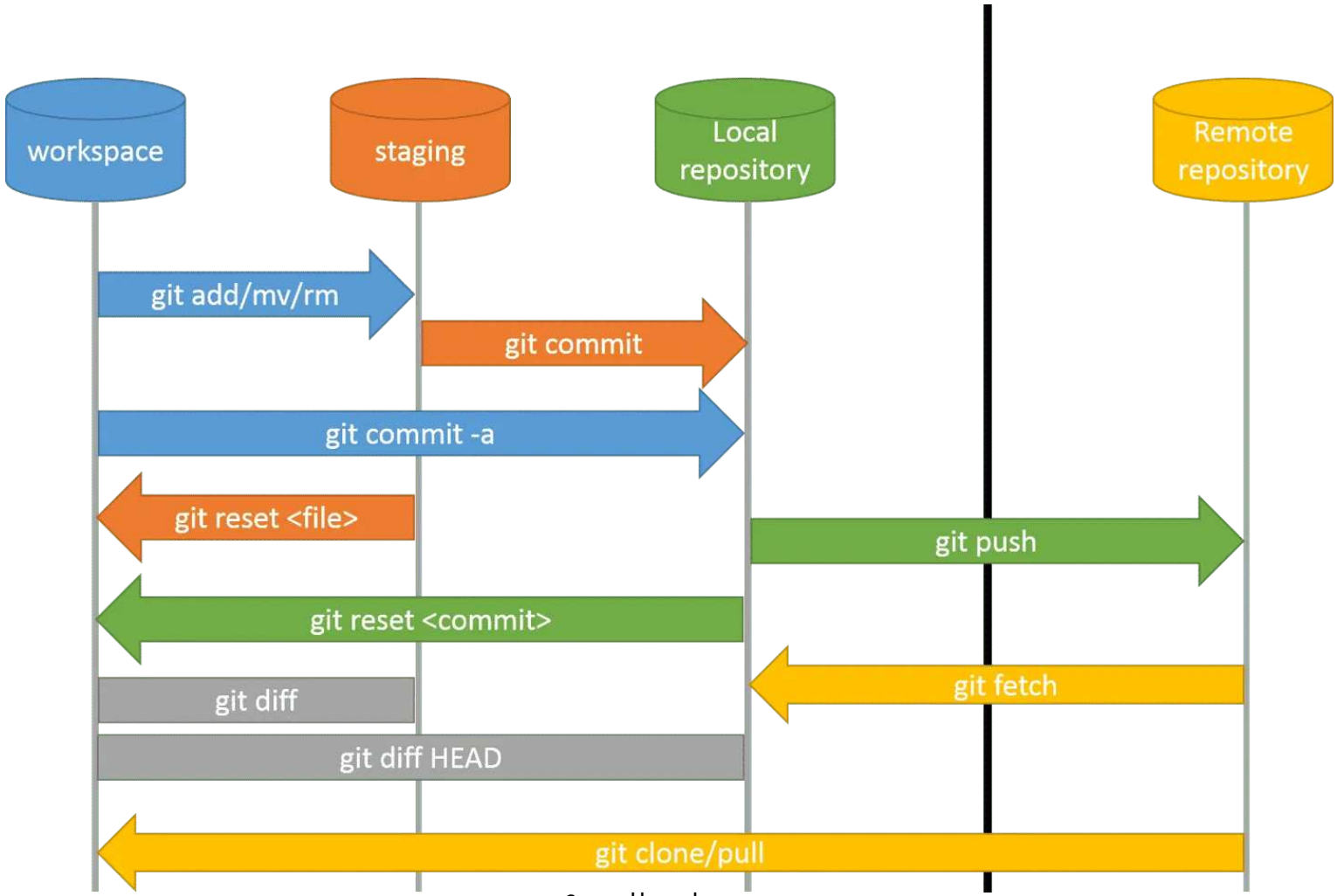
- A file is called a blob (Binary Large Object)
- A directory is a tree
- The top-level tree is a snapshot called commit
- An object is a blob, tree, or commit:
- The predecessor of a snapshot is the commit's parent
- Tracked repositories are called remotes



- The version history is a directed acyclic graph (DAG) of commit
- Each object is addressed via their SHA-1 *hash value*

$\Rightarrow$ Commits are *immutable*

Git Workflow



Source: blog.podrezo.com

Important takeaway messages of this chapter

- RIOT as an IoT can serve as a case study for a distributed system
- Distributed development requires particular tooling – and distributed development tools like DVCS
- Git is probably the most popular example for a DVCS