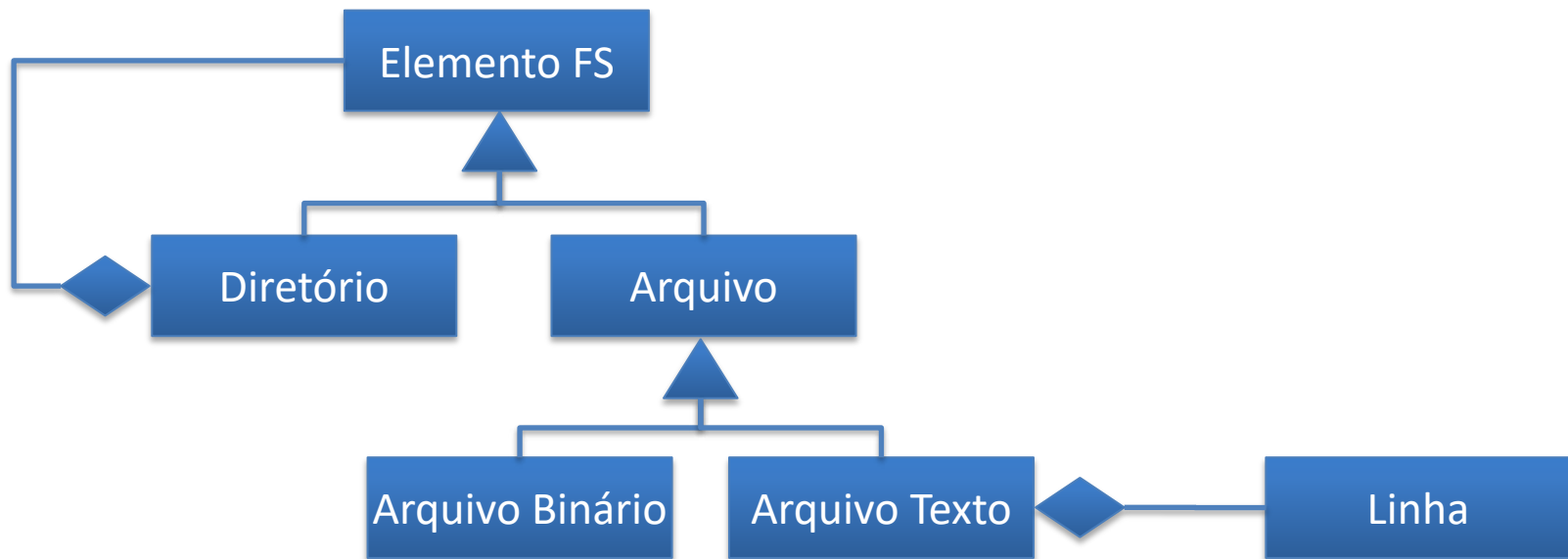


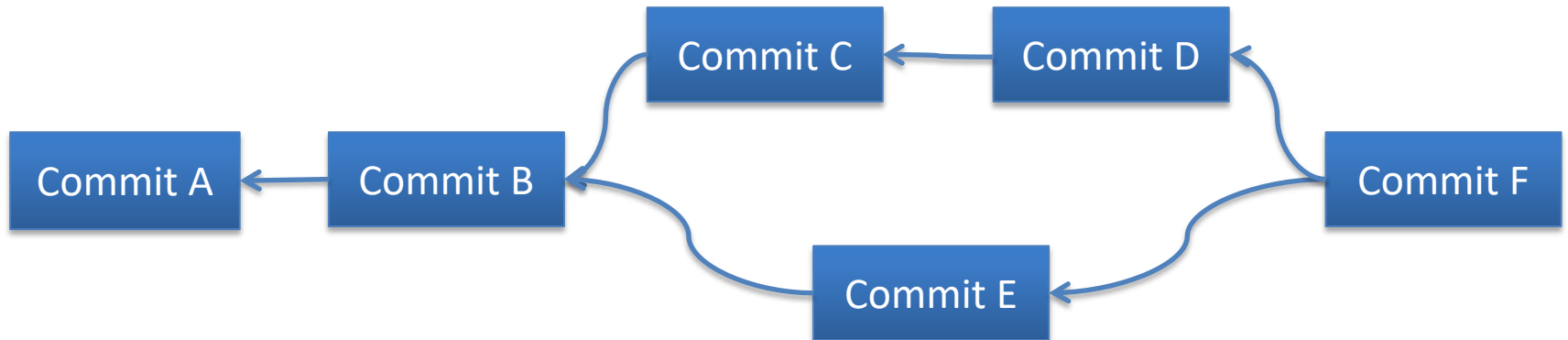
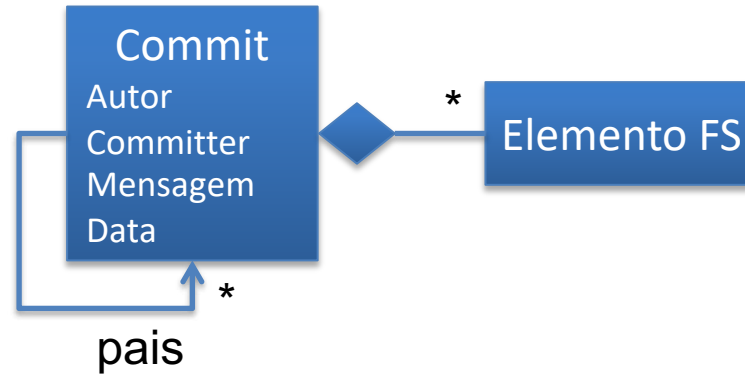
Git



O que é versionado?



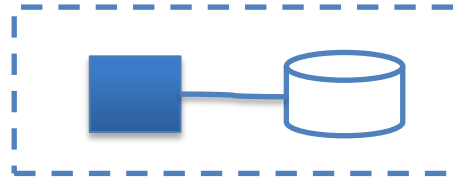
Como é versionado?



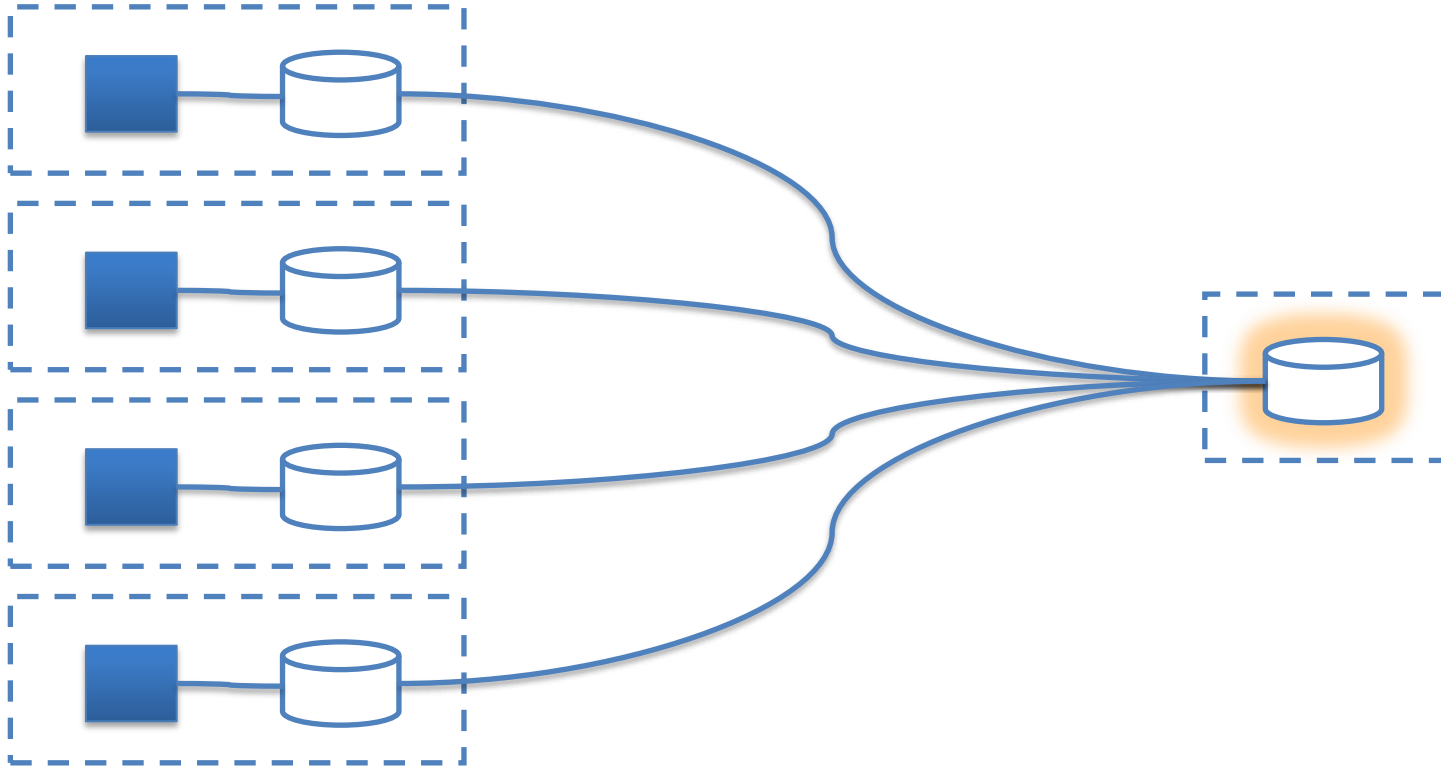
Formas de adoção

- Apesar de ser peer-to-peer, normalmente é definido um “workflow” para adoção de DVCS em função de características do projeto
 - Individual
 - Cliente-servidor
 - Gerente de integração
 - Ditador/tenentes

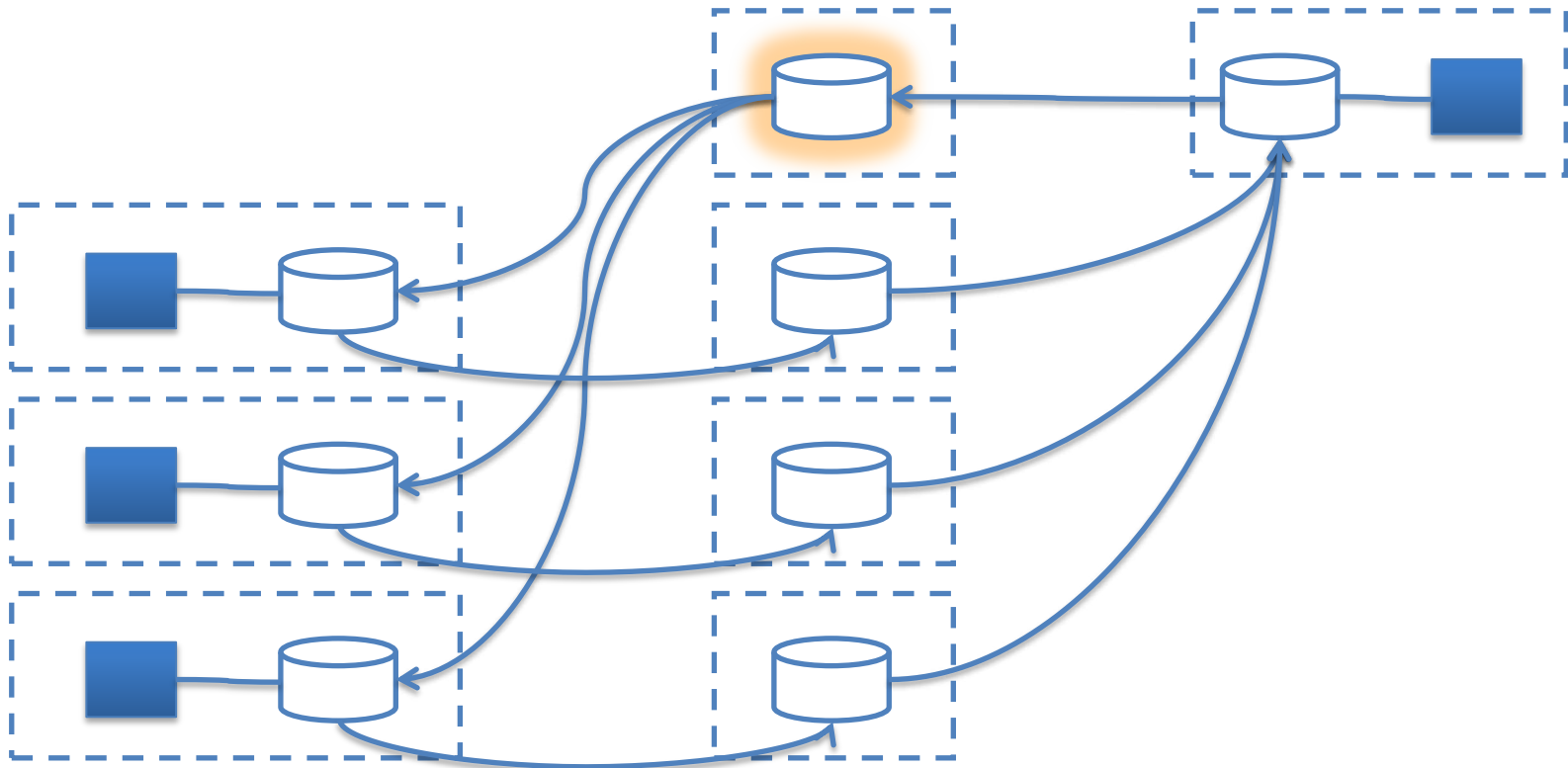
Individual



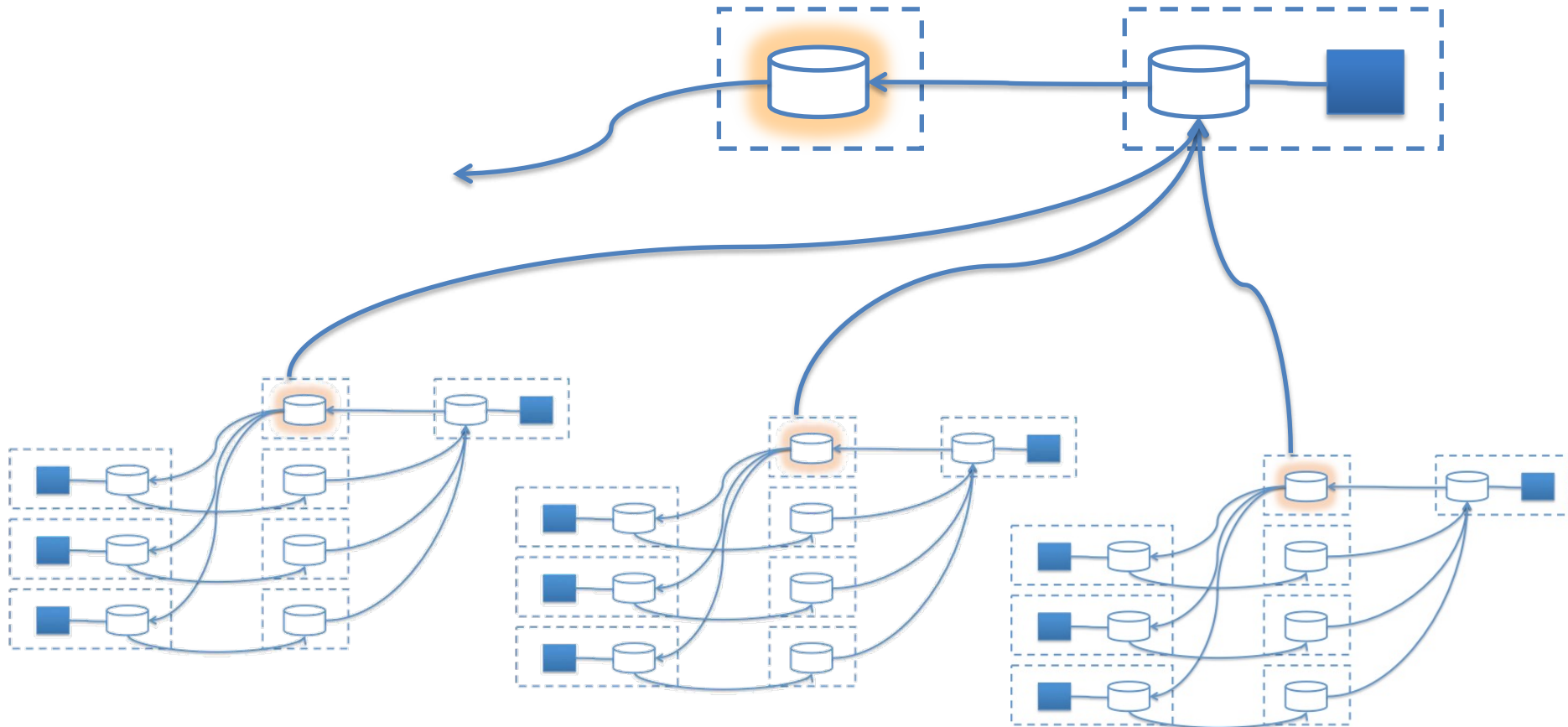
Cliente-servidor



Gerente de integração



Ditador/tenentes



Passo a passo

- Vamos utilizar o Git gradualmente em diferentes situações
 - Conceitos básicos
 - Repositório local
 - Inspeccionando mudanças
 - Demarcando versões especiais
 - Repositório local com ramos
 - Repositório remoto
 - Múltiplos repositórios remotos

Conceitos básicos: *help*!

- `git help`
 - Oferece ajuda geral sobre o git
- `git help <comando>`
 - Oferece ajuda sobre um comando específico do git
- Demais comandos dão dicas do que pode ser feito (leia com atenção as saídas dos comandos!)

Conceitos básicos: quem sou eu?

- `git config --global user.name <seu nome>`
 - Configura o nome do usuário
- `git config --global user.email <seu email>`
 - Configura o email do usuário

Conceitos básicos: *staging area*

- Área onde são colocados os arquivos que pretendemos enviar para o repositório



Conceitos básicos: *commit* id

- Cada sistema de controle de versão usa uma estratégia diferente para identificar *commits*
 - Número sequencial por arquivo (CVS)
 - Número sequencial por repositório (Subversion)
 - Hash (Git e Mercurial)

Conceitos básicos: apelidos

- A versão base do seu espaço de trabalho
 - *HEAD*
- O ramo principal do seu repositório
 - *master ou main*
- O repositório do qual seu repositório foi clonado
 - *origin*

Repositório local

- `git init <nome>`
 - Cria um repositório Git no diretório
- `git add`
 - Adiciona um arquivo na *staging area* para ser enviado ao repositório no próximo *commit*
- `git commit -m <mensagem>`
 - Envia os arquivos que estão na *staging area* para o repositório

Inspecionando mudanças

- `git status`
 - Inspeciona o espaço de trabalho
- `git log [--graph] [--decorate=short] [--name-status]`
 - Inspeciona o histórico do repositório local
- `git show`
 - Inspeciona um *commit*
- `git diff`
 - Compara o espaço de trabalho com a staging area ou com alguma versão do repositório

Interface gráfica

- É possível fazer todos esses passos de forma visual
- Dentre várias ferramentas, vamos praticar com...



Demarcando versões especiais

- `git tag`
 - Lista os rótulos existentes
- `git tag <nome do rótulo> [commit id]`
 - Cria um rótulo sobre um dado commit (HEAD por default)
- `git tag -d <nome do rótulo>`
 - Remove um rótulo

Repositório local com ramos

- `git branch --all -v`
 - Lista os ramos existentes no repositório
- `git branch <nome do ramo>`
 - Cria um ramo à partir da versão indicada no HEAD
- `git branch -d <nome do ramo>`
 - Remove um ramo
- `git checkout <commit id ou nome do ramo>`
 - Troca a versão base do espaço de trabalho
- `git merge <nome do ramo>`
 - Combina um ramo com o ramo corrente

Repositório remoto

- `git clone <url> <diretório>`
 - Cria um repositório local copiando o histórico de um repositório remoto
- `git pull`
 - Atualiza o repositório local e o espaço de trabalho em relação a um repositório remoto
- `git push`
 - Atualiza o repositório remoto em relação ao repositório local

Múltiplos repositórios remotos

- `git remote -v`
 - Listar os repositórios remotos cadastrados
- `git remote add <nome> <url>`
 - Adiciona um novo repositório remoto
- `git remote remove <nome>`
 - Remove um repositório remoto existente

Principal referência bibliográfica

- Chacon, S. Pro Git. Apress, 1ª edição, 2009.

Git

