1️⃣Build Process

✔ Build: Compile code into an executable.

✔ Deploy: Run the executable on the system.

Initially, compile via the command line:

```
g++ config_parser.cc config_parser_main.cc
-std=c++0x -g -Wall -o config_parser
```

Use scripts to automate the build process, especially when different compilers or platforms are involved.

✔ Example of a cross-platform build script (bash):

```
#!/bin/bash
case uname in
  Linux) g++ config_parser.cc -std=c++0x -g
-Wall -o config_parser;;
  Darwin) clang++ config_parser.cc
-std=c++11 -g -Wall -stdlib=libc++ -o
config_parser;;
  *) echo "Unknown OS";;
esac
```

2️⃣Makefiles & Tools

✔ Makefile: More structured than bash scripts, better for complex projects.

```
CXXFLAGS=-std=c++11 -Wall -Werror
config_parser_main: config_parser_main.o
config_parser.o
    $(LINK.cc) $^ $(LDLIBS) -o $@
config_parser.o: config_parser.cc
config_parser.h
    $(CXX) $(CXXFLAGS) -c $<
```

✔ Autoconf + Automake generates Makefiles automatically.

✔ Bazel: For large-scale builds (e.g., multi-language projects).: “$ bazel build //:my_app”

3️⃣Google’s Build Scalability

✔ Google’s Scale:

~100k developers, 2B lines of code, 15M lines changed weekly.
Builds for large teams with multiple changes every second.

✔ Build Scalability Challenges:

Separate repositories (Multi-repo): Faster builds as you only compile your code, but require manual fixes after library updates.
Mono-repo: All code in a single repository, but slower builds due to the need to build the entire codebase.

✔ Google’s Solution:

Small changes per commit, 90% cache hit rate, and distributed across 10k CPU cores.
CitC (Cloud-in-the-Cloud) stores changes, compiles remotely, and reuses builds, significantly improving speed.

✔ Google's solution for build scalability includes:

Small Commits: Developers make small changes to the code, so only a small part of the codebase is rebuilt, speeding up the process.

Cache Usage: Google achieves a 90% cache hit rate, reusing previous builds instead of rebuilding the same code, saving time and resources.

Cloud-Based Compilation: Changes are sent to the cloud for compilation, making the process faster. Developers work with the cloud as if it were local through a FUSE file system.

Parallel Processing: Google uses 10,000 CPU cores and 50 TB of RAM to handle builds in parallel, drastically improving build speed.

📌 Key Takeaways:

Build and deployment should be automated and repeatable with scripts or Makefiles.

Use scalable build systems like Bazel or Google’s mono-repo strategy to manage large-scale projects effectively.

Caching and cloud solutions can greatly reduce build times and handle large codebases efficiently.